Oracle® Fusion Middleware

Reference for Oracle GoldenGate

18c (18.1.0)
E95987-06
August 2020

Oracle Fusion Middleware Reference for Oracle GoldenGate, 18c (18.1.0)

E95987-06

Copyright © 2013, 2020, Oracle and/or its affiliates.

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

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government’s use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware 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 that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about 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 unless otherwise set forth in an applicable agreement between you and Oracle. 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, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience xiii
Documentation Accessibility xiii
Related Information xiv
Conventions xiv

1 Prerequisites for Using Oracle GoldenGate

Part I Oracle GoldenGate Common Components

2 Oracle GoldenGate Parameters

2.31 COMMIT_SERIALIZATION 2-6
2.1 Summary of Oracle GoldenGate Parameters 2-6
2.1.1 Summary of GLOBALS Parameters 2-7
2.1.2 Summary of Parameters Common to Extract and Replicat 2-9
2.1.3 Summary of Extract Parameters 2-13
2.1.4 Summary of Replicat Parameters 2-15
2.1.5 Summary of Wildcard Exclusion Parameters 2-19
2.1.6 Summary of DEFGEN Parameters 2-19
2.1.7 Summary of DDL Parameters 2-20
2.2 ABORTDISCARDRECS 2-21
2.3 ALLOCFILES 2-21
2.4 ALLOWDUPTARGETMAP | NOALLOWDUPTARGETMAP 2-22
2.5 ALLOWINVISIBLEINDEXKEYS 2-23
2.6 ALLOWNULLABLEKEYS | NOALLOWNULLABLEKEYS 2-24
2.7 ALLOWNONVALIDATEDKEYS 2-24
2.8 ALLOWNOOPUPDATES | NOALLOWNOOPUPDATES 2-25
2.9 ALLOWOUTPUTDIR 2-27
2.10 APPLYNOOPUPDATES | NOAPPLYNOOPUPDATES 2-27

2.11 APPLY_PARALLELISM | MAX_APPLY_PARALLELISM | MIN_APPLY_PARALLELISM 2-28
2.12 ASCIITOEBCDIC 2-28
2.13 ASSUMETARGETDEFS 2-29
2.14 BATCHSQL 2-29
2.15 BEGIN 2-34
2.16 BLOBMEMORY 2-34
2.17 BR 2-34
2.18 BULKLOAD 2-41
2.19 CACHEMGR 2-42
2.20 CATALOGEXCLUDE 2-44
2.21 CHARMAP 2-45
2.22 CHECKPARAMS 2-46
2.23 CHECKPOINTSECS 2-46
2.24 CHECKPOINTTABLE 2-47
2.25 CHUNK_SIZE 2-48
2.26 CMDTRACE 2-48
2.27 COLCHARSET 2-49
2.28 COLMATCH 2-50
2.29 COMPRESSDELETES | NOCOMPRESSDELETES 2-52
2.30 COMPRESSUPDATES | NOCOMPRESSUPDATES 2-53
2.32 COORDSTATINTERVAL 2-54
2.33 COORDTIMER 2-54
2.34 CREDENTIALSTORELOCATION 2-55
2.35 CRYPTOENGINE 2-55
2.36 CUSEREXIT 2-56
2.37 DBOPTIONS 2-58
2.38 DDL 2-67
2.39 DDLERROR 2-75
2.40 DDLOPTIONS 2-78
2.41 DDLSUBST 2-86
2.42 DDLRULEHINT 2-88
2.43 DDLTABLE 2-89
2.44 DECRYPTTRAIL 2-89
2.45 DEFERAPPLYINTERVAL 2-90
2.46 DEFSFILE 2-91
2.47 DISCARDFILE | NODISCARDFILE 2-93
2.48 DISCARDROLLOVER 2-95
2.49 DYNAMICRESOLUTION | NODYNAMICRESOLUTION 2-96
2.50 EBCDICTOASCII 2-97
2.51 ENABLEMONITORING 2-97

2.52 ENABLE_HEARTBEAT_TABLE | DISABLE_HEARTBEAT_TABLE 2-98
2.53 ENCRYPTTRAIL | NOENCRYPTTRAIL 2-99
2.54 END 2-101
2.55 EOFDELAY | EOFDELAYCSECS 2-102
2.56 EXCLUDEHIDDENCOLUMNS 2-103
2.57 EXCLUDETAG 2-103
2.58 EXCLUDEWILDCARDOBJECTSONLY 2-104
2.59 EXTFILE 2-105
2.60 EXTRACT 2-106
2.61 EXTTRAIL 2-107
2.62 FETCHOPTIONS 2-109
2.63 FETCHUSERID 2-112
2.64 FETCHUSERIDALIAS 2-115
2.65 FILTERDUPS | NOFILTERDUPS 2-116
2.66 FLUSHSECS | FLUSHCSECS 2-117
2.67 FUNCTIONSTACKSIZE 2-118
2.68 GETAPPLOPS | IGNOREAPPLOPS 2-119
2.69 GETDELETES | IGNOREDELETES 2-120
2.70 GETINSERTS | IGNOREINSERTS 2-120
2.71 GETREPLICATES | IGNOREREPLICATES 2-121
2.72 GETTRUNCATES | IGNORETRUNCATES 2-122
2.73 GETUPDATEAFTERS | IGNOREUPDATEAFTERS 2-124
2.74 GETUPDATEBEFORES | IGNOREUPDATEBEFORES 2-124
2.75 GETUPDATES | IGNOREUPDATES 2-126
2.76 GGSCHEMA 2-126
2.77 GROUPTRANSOPS 2-127
2.78 HANDLECOLLISIONS | NOHANDLECOLLISIONS 2-128
2.79 HANDLETPKUPDATE 2-133
2.80 HAVEUDTWITHNCHAR 2-134
2.81 HEARTBEATTABLE 2-134
2.82 INCLUDE 2-135
2.83 INSERTALLRECORDS 2-135
2.84 INSERTAPPEND | NOINSERTAPPEND 2-137
2.85 INSERTDELETES | NOINSERTDELETES 2-138
2.86 INSERTMISSINGUPDATES | NOINSERTMISSINGUPDATES 2-138
2.87 INSERTUPDATES | NOINSERTUPDATES 2-139
2.88 LIST | NOLIST 2-140
2.89 LOBMEMORY 2-140
2.90 LOGALLSUPCOLS 2-143
2.91 LOOK_AHEAD_TRANSACTIONS 2-144
2.92 MACRO 2-144

2.93 MACROCHAR 2-146
2.94 MAP for Extract 2-147
2.95 MAP 2-148
2.96 MAPALLCOLUMNS| NOMAPALLCOLUMNS 2-148
2.97 MAP_PARALLELISM 2-149
2.98 MAPEXCLUDE 2-149
2.99 MAPINVISIBLECOLUMNS | NOMAPINVISIBLECOLUMNS 2-151
2.100 MARKERTABLE 2-152
2.101 MARKERTABLE 2-153
2.102 MAXDISCARDRECS 2-153
2.103 MAXGROUPS 2-154
2.104 MAXSQLSTATEMENTS 2-154
2.105 MAXTRANSOPS 2-155
2.106 MGRSERVNAME 2-156
2.107 NAMECCSID 2-157
2.108 NAMEMATCH parameters 2-157
2.109 NOCATALOG 2-158
2.110 NODUPMSGSUPPRESSION 2-158
2.111 NUMFILES 2-159
2.112 OBEY 2-160
2.113 OUTPUTFILEUMASK 2-160
2.114 OUTPUTFORMAT 2-161
2.115 OVERRIDEDUPS | NOOVERRIDEDUPS 2-165
2.116 PTKCAPTUREPROCSTATS 2-166
2.117 PTKMONITORFREQUENCY 2-167
2.118 PRESERVETARGETTIMEZONE 2-167
2.119 PROCEDURE 2-168
2.120 RECOVERYOPTIONS 2-168
2.121 REPERROR 2-169
2.122 REPFETCHEDCOLOPTIONS 2-175
2.123 REPLACEBADCHAR 2-178
2.124 REPLACEBADNUM 2-179
2.125 REPLICAT 2-180
2.126 REPORT 2-180
2.127 REPORTCOUNT 2-182
2.128 REPORTROLLOVER 2-183
2.129 REPOUDPPORT 2-184
2.130 RESTARTCOLLISIONS | NORESTARTCOLLISIONS 2-185
2.131 RMTFILE 2-185
2.132 RMTHOST 2-187
2.133 RMTHOSTOPTIONS 2-192

2.134 RMTTASK 2-196
2.135 RMTTRAIL 2-197
2.136 ROLLOVER 2-199
2.137 SCHEMAEXCLUDE 2-201
2.138 SEQUENCE 2-202
2.139 SESSIONCHARSET 2-204
2.140 SETENV 2-205
2.141 SHOWSYNTAX 2-207
2.142 SOURCEDB 2-208
2.143 SOURCECATALOG 2-209
2.144 SOURCECHARSET 2-210
2.145 SOURCEDEFS 2-212
2.146 SOURCEISTABLE 2-213
2.147 SOURCETIMEZONE 2-214
2.148 SPACESTONULL | NOSPACESTONULL 2-215
2.149 SPECIALRUN 2-216
2.150 SPLIT_TRANS_REC 2-217
2.151 SQLDUPERR 2-217
2.152 SQLEXEC 2-218
2.153 STATOPTIONS 2-229
2.154 TABLE | MAP 2-231
2.155 TABLE for DEFGEN 2-276
2.156 TABLE for Replicat 2-277
2.157 TABLEEXCLUDE 2-279
2.158 TARGETDB 2-280
2.159 TARGETDEFS 2-281
2.160 TCPSOURCETIMER | NOTCPSOURCETIMER 2-282
2.161 THREADOPTIONS 2-282
2.162 TRACE | TRACE2 2-284
2.163 TRACETABLE | NOTRACETABLE 2-285
2.164 TRAILBYTEORDER 2-286
2.165 TRAILCHARSET 2-288
2.166 TRAILCHARSETASCII 2-289
2.167 TRAILCHARSETEBCDIC 2-289
2.168 TRAIL_SEQLEN_6D | TRAIL_SEQLEN_9D 2-290
2.169 TRANLOGOPTIONS 2-291
2.170 TRANSACTIONTIMEOUT 2-317
2.171 TRANSMEMORY 2-318
2.172 TRIMSPACES | NOTRIMSPACES 2-321
2.173 TRIMVARSPACES | NOTRIMVARSPACES 2-322
2.174 UPDATEDELETES | NOUPDATEDELETES 2-323

2.175 UPDATEINSERTS | NOUPDATEINSERTS 2-324
2.176 UPDATERECORDFORMAT 2-324
2.177 UPREPORT 2-326
2.178 USE_TRAILDEFS | NO_USE_TRAILDEFS 2-326
2.179 USEDEDICATEDCOORDINATIONTHREAD 2-327
2.180 USEIPV4 2-328
2.181 USEIPV6 2-328
2.182 USERID | NOUSERID 2-328
2.183 USERIDALIAS 2-334
2.184 VAM 2-337
2.185 VARWIDTHNCHAR | NOVARWIDTHNCHAR 2-338
2.186 WALLETLOCATION 2-338
2.187 WARNLONGTRANS 2-339
2.188 WARNRATE 2-340
2.189 WILDCARDRESOLVE 2-341
2.190 XAGENABLE 2-342
2.191 Y2KCENTURYADJUSTMENT | NOY2KCENTURYADJUSTMENT 2-342

3 Column Conversion Functions

3.1 Summary of Column-Conversion Functions 3-2
3.2 @RANGE 3-4
3.3 AFTER 3-6
3.4 BEFORE 3-6
3.5 BEFOREAFTER 3-6
3.6 BINARY 3-7
3.7 BINTOHEX 3-7
3.8 CASE 3-8
3.9 COLSTAT 3-9
3.10 COLTEST 3-9
3.11 COMPUTE 3-10
3.12 DATE 3-11
3.13 DATEDIFF 3-15
3.14 DATENOW 3-15
3.15 DDL 3-15
3.16 EVAL 3-16
3.17 GETENV 3-17
3.18 GETVAL 3-34
3.19 HEXTOBIN 3-36
3.20 HIGHVAL | LOWVAL 3-36
3.21 IF 3-37

3.22 NUMBIN 3-38
3.23 NUMSTR 3-38
3.24 OGG_SHA1 3-39
3.25 STRCAT 3-39
3.26 STRCMP 3-39
3.27 STRCMPNULL 3-40
3.28 STREQ 3-40
3.29 STREQNULL 3-41
3.30 STREXT 3-41
3.31 STRFIND 3-42
3.32 STRLEN 3-43
3.33 STRLTRIM 3-43
3.34 STRNCAT 3-44
3.35 STRNCMP 3-44
3.36 STRNUM 3-45
3.37 STRRTRIM 3-46
3.38 STRSUB 3-46
3.39 STRTRIM 3-47
3.40 STRUP 3-48
3.41 TOKEN 3-48
3.42 VALONEOF 3-49

4 User Exit Functions

4.3 Using EXIT_CALL_TYPE 4-2
4.51 SET_SESSION_CHARSET 4-3
4.1 Calling a User Exit 4-5
4.2 Summary of User Exit Functions 4-5
4.4 Using EXIT_CALL_RESULT 4-5
4.5 Using EXIT_PARAMS 4-6
4.6 Using ERCALLBACK 4-6
4.7 Function Codes 4-8
4.8 COMPRESS_RECORD 4-11
4.9 DECOMPRESS_RECORD 4-12
4.10 GET_BASE_OBJECT_NAME 4-14
4.11 GET_BASE_OBJECT_NAME_ONLY 4-15
4.12 GET_BASE_SCHEMA_NAME_ONLY 4-17
4.13 GET_BEFORE_AFTER_IND 4-18
4.14 GET_CATALOG_NAME_ONLY 4-19
4.15 GET_COL_METADATA_FROM_INDEX 4-21
4.16 GET_COL_METADATA_FROM_NAME 4-23

4.17 GET_COLUMN_INDEX_FROM_NAME 4-26
4.18 GET_COLUMN_NAME_FROM_INDEX 4-27
4.19 GET_COLUMN_VALUE_FROM_INDEX 4-29
4.20 GET_COLUMN_VALUE_FROM_NAME 4-32
4.21 GET_DATABASE_METADATA 4-36
4.22 GET_DDL_RECORD_PROPERTIES 4-37
4.23 GETENV 4-40
4.24 GET_ENV_VALUE 4-57
4.25 GET_ERROR_INFO 4-58
4.26 GET_GMT_TIMESTAMP 4-59
4.27 GET_MARKER_INFO 4-60
4.28 GET_OBJECT_NAME 4-62
4.29 GET_OBJECT_NAME_ONLY 4-63
4.30 GET_OPERATION_TYPE 4-65
4.31 GET_POSITION 4-66
4.32 GET_RECORD_BUFFER 4-67
4.33 GET_RECORD_LENGTH 4-69
4.34 GET_RECORD_TYPE 4-70
4.35 GET_SCHEMA_NAME_ONLY 4-72
4.36 GET_SESSION_CHARSET 4-73
4.37 GET_STATISTICS 4-74
4.38 GET_TABLE_COLUMN_COUNT 4-76
4.39 GET_TABLE_METADATA 4-77
4.40 GET_TABLE_NAME 4-79
4.41 GET_TABLE_NAME_ONLY 4-81
4.42 GET_TIMESTAMP 4-82
4.43 GET_TRANSACTION_IND 4-83
4.44 GET_USER_TOKEN_VALUE 4-84
4.45 OUTPUT_MESSAGE_TO_REPORT 4-86
4.46 RESET_USEREXIT_STATS 4-86
4.47 SET_COLUMN_VALUE_BY_INDEX 4-87
4.48 SET_COLUMN_VALUE_BY_NAME 4-89
4.49 SET_OPERATION_TYPE 4-92
4.50 SET_RECORD_BUFFER 4-93
4.52 SET_TABLE_NAME 4-94

Part II Oracle GoldenGate Microservices Architecture

5 Oracle GoldenGate AdminClient Commands

5.1 Admin Client Commands 5-1

Part III Oracle GoldenGate Classic Architecture

6 Collector Parameters

6.1 Overview of the Collector Process 6-1
6.2 Summary of Collector Parameters 6-2
6.3 -B 6-2
6.4 -b 6-3
6.5 -cp 6-3
6.6 -d 6-3
6.7 -E 6-4
6.8 -e 6-4
6.9 -ENCRYPT 6-5
6.10 -f 6-5
6.11 -g 6-5
6.12 -h 6-6
6.13 -k 6-6
6.14 -KEYNAME 6-6
6.15 -l 6-6
6.16 -m 6-7
6.17 -P 6-7
6.18 -p 6-7
6.19 -R 6-8
6.20 -x 6-8

7 Manager Parameters

7.1 Summary of Manager Parameters 7-1
7.2 ACCESSRULE 7-3
7.3 AUTORESTART 7-5
7.4 AUTOSTART 7-6
7.5 BOOTDELAYMINUTES 7-7
7.6 CHARSET 7-8
7.7 CHECKMINUTES 7-9
7.8 COMMENT | -- 7-10
7.9 DOWNREPORT 7-11
7.10 DYNAMICPORTLIST 7-11

7.11 MONITORING_HEARTBEAT_TIMEOUT 7-12
7.12 LAGCRITICAL 7-13
7.13 LAGINFO 7-13
7.14 LAGREPORT 7-14
7.15 PURGEMARKERHISTORY 7-15
7.16 PORT 7-16
7.17 SOURCEDB 7-17
7.18 PURGEDDLHISTORY | PURGEDDLHISTORYALT 7-18
7.19 PURGEOLDEXTRACTS for Extract and Replicat 7-20
7.20 PURGEOLDEXTRACTS for Manager 7-20
7.21 PURGEOLDTASKS 7-24
7.22 STARTUPVALIDATIONDELAY[CSECS] 7-25
7.23 USERIDALIAS 7-25
7.24 VERIDATAREPORTAGE 7-28

8 Oracle GoldenGate Native Commands

8.1 Summary of Oracle GoldenGate IBM i Native Commands 8-1
8.2 checkprm 8-3
8.3 convchk 8-5
8.4 defgen 8-6
8.5 extract 8-7
8.6 install 8-8
8.7 keygen 8-9
8.8 logdump 8-10
8.9 mgr 8-10
8.10 replicat 8-11

9 Oracle GoldenGate GGSCI Commands

9.1 GGSCI Command Interface 9-1

Preface

This guide contains reference information, with usage and syntax guidelines, for:

- Oracle GoldenGate GGSCI commands.
- Oracle GoldenGate configuration parameters.
- Oracle GoldenGate column-conversion functions.
- Oracle GoldenGate user exit functions.
- Oracle GoldenGate AdminClient commands
- Oracle GoldenGate commands, parameters, and functions for heterogeneous databases

> **Note:**
>
> Oracle GoldenGate 18c (18.1.0) is not released for SQL Server and DB2 for i. However, the documentation may include information associated with these databases.

- Audience
- Documentation Accessibility
- Related Information
- Conventions

Audience

This guide is intended for the person or persons who are responsible for operating Oracle GoldenGate and maintaining its performance. This audience typically includes, but is not limited to, systems administrators and database administrators.

Documentation Accessibility

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

Accessible Access to Oracle Support

Oracle customers who have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/`

topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Information

The Oracle GoldenGate Product Documentation Libraries are found at

https://docs.oracle.com/en/middleware/goldengate/index.html

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

Oracle GoldenGate A-Team Chronicles

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, such as "From the File menu, select **Save**." Boldface also is used for terms defined in text or in the glossary.
italic *italic*	Italic type indicates placeholder variables for which you supply particular values, such as in the parameter statement: TABLE *table_name*. Italic type also is used for book titles and emphasis.
monospace MONOSPACE	Monospace type indicates code components such as user exits and scripts; the names of files and database objects; URL paths; and input and output text that appears on the screen. Uppercase monospace type is generally used to represent the names of Oracle GoldenGate parameters, commands, and user-configurable functions, as well as SQL commands and keywords.
UPPERCASE	Uppercase in the regular text font indicates the name of a utility unless the name is intended to be a specific case.
{ }	Braces within syntax enclose a set of options that are separated by pipe symbols, one of which must be selected, for example: {*option1* | *option2* | *option3*}.
[ ]	Brackets within syntax indicate an optional element. For example in this syntax, the SAVE clause is optional: CLEANUP REPLICAT *group_name* [, SAVE *count*]. Multiple options within an optional element are separated by a pipe symbol, for example: [*option1* | *option2*].

1

Prerequisites for Using Oracle GoldenGate

Before using the Oracle GoldenGate commands, parameters, and functions, prepare and configure your database.

The product set enables high availability solutions, real-time data integration, transactional change data capture, data replication, transformations, and verification between operational and analytical enterprise systems. Oracle GoldenGate 18c brings extreme performance with simplified configuration and management, support for cloud environments, expanded heterogeneity, and enhanced security.

You can use the following supported heterogeneous databases with Oracle GoldenGate.

- DB2 LUW
-
- DB2 for z/OS
- MySQL
- Teradata

> **Note:**
>
> Oracle GoldenGate 18c (18.1.0) is not released for SQL Server and DB2 for i. However, the documentation may include information associated with these databases.

Part I
Oracle GoldenGate Common Components

There are a number of data replication processes that are common to both Oracle GoldenGate architectures.

This part describes the commands, parameters, and options that you can use with the Oracle GoldenGate. which are common to both product architectures..

Topics:

- Oracle GoldenGate Parameters
- Column Conversion Functions
- User Exit Functions

2

Oracle GoldenGate Parameters

This chapter contains summaries of the Oracle GoldenGate parameters that control processing, followed by detailed descriptions of each parameter in alphabetical order. For instructions on creating, changing, and storing Oracle GoldenGate parameter files, see Administering Oracle GoldenGate for Windows and UNIX.

- Summary of Oracle GoldenGate Parameters
- ABORTDISCARDRECS
- ALLOCFILES
- ALLOWDUPTARGETMAP | NOALLOWDUPTARGETMAP
- ALLOWINVISIBLEINDEXKEYS
- ALLOWNULLABLEKEYS | NOALLOWNULLABLEKEYS
- ALLOWNONVALIDATEDKEYS
- ALLOWNOOPUPDATES | NOALLOWNOOPUPDATES
- ALLOWOUTPUTDIR
- APPLYNOOPUPDATES | NOAPPLYNOOPUPDATES
- APPLY_PARALLELISM | MAX_APPLY_PARALLELISM | MIN_APPLY_PARALLELISM
- ASCIITOEBCDIC
- ASSUMETARGETDEFS
- BATCHSQL
- BEGIN
- BLOBMEMORY
- BR
- BULKLOAD
- CACHEMGR
- CATALOGEXCLUDE
- CHARMAP
- CHECKPARAMS
- CHECKPOINTSECS
- CHECKPOINTTABLE
- CHUNK_SIZE
- CMDTRACE
- COLCHARSET
- COLMATCH

- COMPRESSDELETES | NOCOMPRESSDELETES
- COMPRESSUPDATES | NOCOMPRESSUPDATES
- COMMIT_SERIALIZATION
- COORDSTATINTERVAL
- COORDTIMER
- CREDENTIALSTORELOCATION
- CRYPTOENGINE
- CUSEREXIT
- DBOPTIONS
- DDL
- DDLERROR
- DDLOPTIONS
- DDLSUBST
- DDLRULEHINT
- DDLTABLE
- DECRYPTTRAIL
- DEFERAPPLYINTERVAL
- DEFSFILE
- DISCARDFILE | NODISCARDFILE
- DISCARDROLLOVER
- DYNAMICRESOLUTION | NODYNAMICRESOLUTION
- EBCDICTOASCII
- ENABLEMONITORING
- ENABLE_HEARTBEAT_TABLE | DISABLE_HEARTBEAT_TABLE
- ENCRYPTTRAIL | NOENCRYPTTRAIL
- END
- EOFDELAY | EOFDELAYCSECS
- EXCLUDEHIDDENCOLUMNS
- EXCLUDETAG
- EXCLUDEWILDCARDOBJECTSONLY
- EXTFILE
- EXTRACT
- EXTTRAIL
- FETCHOPTIONS
- FETCHUSERID
- FETCHUSERIDALIAS
- FILTERDUPS | NOFILTERDUPS

- FLUSHSECS | FLUSHCSECS
- FUNCTIONSTACKSIZE
- GETAPPLOPS | IGNOREAPPLOPS
- GETDELETES | IGNOREDELETES
- GETINSERTS | IGNOREINSERTS
- GETREPLICATES | IGNOREREPLICATES
- GETTRUNCATES | IGNORETRUNCATES
- GETUPDATEAFTERS | IGNOREUPDATEAFTERS
- GETUPDATEBEFORES | IGNOREUPDATEBEFORES
- GETUPDATES | IGNOREUPDATES
- GGSCHEMA
- GROUPTRANSOPS
- HANDLECOLLISIONS | NOHANDLECOLLISIONS
- HANDLETPKUPDATE
- HAVEUDTWITHNCHAR
- HEARTBEATTABLE
- INCLUDE
- INSERTALLRECORDS
- INSERTAPPEND | NOINSERTAPPEND
- INSERTDELETES | NOINSERTDELETES
- INSERTMISSINGUPDATES | NOINSERTMISSINGUPDATES
- INSERTUPDATES | NOINSERTUPDATES
- LIST | NOLIST
- LOBMEMORY
- LOGALLSUPCOLS
- LOOK_AHEAD_TRANSACTIONS
- MACRO
- MACROCHAR
- MAP for Extract
- MAP
- MAPALLCOLUMNS| NOMAPALLCOLUMNS
- MAP_PARALLELISM
- MAPEXCLUDE
- MAPINVISIBLECOLUMNS | NOMAPINVISIBLECOLUMNS
- MARKERTABLE
- MARKERTABLE
- MAXDISCARDRECS

- MAXGROUPS
- MAXSQLSTATEMENTS
- MAXTRANSOPS
- MGRSERVNAME
- NAMECCSID
- NAMEMATCH parameters
- NOCATALOG
- NODUPMSGSUPPRESSION
- NUMFILES
- OBEY
- OUTPUTFILEUMASK
- OUTPUTFORMAT
- OVERRIDEDUPS | NOOVERRIDEDUPS
- PTKCAPTUREPROCSTATS
- PTKMONITORFREQUENCY
- PRESERVETARGETTIMEZONE
- PROCEDURE
- RECOVERYOPTIONS
- REPERROR
- REPFETCHEDCOLOPTIONS
- REPLACEBADCHAR
- REPLACEBADNUM
- REPLICAT
- REPORT
- REPORTCOUNT
- REPORTROLLOVER
- REPOUDPPORT
- RESTARTCOLLISIONS | NORESTARTCOLLISIONS
- RMTFILE
- RMTHOST
- RMTHOSTOPTIONS
- RMTTASK
- RMTTRAIL
- ROLLOVER
- SCHEMAEXCLUDE
- SEQUENCE
- SESSIONCHARSET

- SETENV
- SHOWSYNTAX
- SOURCEDB
- SOURCECATALOG
- SOURCECHARSET
- SOURCEDEFS
- SOURCEISTABLE
- SOURCETIMEZONE
- SPACESTONULL | NOSPACESTONULL
- SPECIALRUN
- SPLIT_TRANS_REC
- SQLDUPERR
- SQLEXEC
- STATOPTIONS
- TABLE | MAP
- TABLE for DEFGEN
- TABLE for Replicat
- TABLEEXCLUDE
- TARGETDB
- TARGETDEFS
- TCPSOURCETIMER | NOTCPSOURCETIMER
- THREADOPTIONS
- TRACE | TRACE2
- TRACETABLE | NOTRACETABLE
- TRAILBYTEORDER
- TRAILCHARSET
- TRAILCHARSETASCII
- TRAILCHARSETEBCDIC
- TRAIL_SEQLEN_6D | TRAIL_SEQLEN_9D
- TRANLOGOPTIONS
- TRANSACTIONTIMEOUT
- TRANSMEMORY
- TRIMSPACES | NOTRIMSPACES
- TRIMVARSPACES | NOTRIMVARSPACES
- UPDATEDELETES | NOUPDATEDELETES
- UPDATEINSERTS | NOUPDATEINSERTS
- UPDATERECORDFORMAT

- UPREPORT
- USE_TRAILDEFS | NO_USE_TRAILDEFS
- USEDEDICATEDCOORDINATIONTHREAD
- USEIPV4
- USEIPV6
- USERID | NOUSERID
- USERIDALIAS
- VAM
- VARWIDTHNCHAR | NOVARWIDTHNCHAR
- WALLETLOCATION
- WARNLONGTRANS
- WARNRATE
- WILDCARDRESOLVE
- XAGENABLE
- Y2KCENTURYADJUSTMENT | NOY2KCENTURYADJUSTMENT

2.31 COMMIT_SERIALIZATION

Valid For

Parallel Replicat

Description

Enables commit FULL serialization mode, which forces transactions to be committed in trail order. See Basic Parameters for Parallel Replicat in *Using Oracle GoldenGate for Oracle Database*.

Default

None

2.1 Summary of Oracle GoldenGate Parameters

This section summarizes the Oracle GoldenGate parameters by module and purpose and includes the following topics:

Topics:

- Summary of GLOBALS Parameters
- Summary of Parameters Common to Extract and Replicat
- Summary of Extract Parameters
- Summary of Replicat Parameters
- Summary of Wildcard Exclusion Parameters
- Summary of DEFGEN Parameters

- Summary of DDL Parameters

2.1.1 Summary of GLOBALS Parameters

The GLOBALS file stores parameters that relate to the Oracle GoldenGate instance as a whole, as opposed to runtime parameters for a specific process.

Table 2-1 All GLOBALS Parameters

Parameter	Description
ALLOWOUTPUTDIR	Use to ALLOWOUTPUTDIR specify the allowed output trail directory (including its subdirectories) through \u000b.
ALLOWNULLABLEKEYS | NOALLOWNULLABLEKEYS	Use ALLOWNULLABLEKEYS to change the key selection logic so that it does not consider a nullable unique key.
CACHEMGR	Manages virtual memory resources.
CHARMAP	Specifies that the character mapping file overrides the character code point mapping.
CHARSET	Specifies a multibyte character set for the process to use instead of the operating system default when reading the parameter file.
CHECKPOINTTABLE	Specifies a default checkpoint table.
CREDENTIALSTORELOCATION	Specifies the location of the Oracle GoldenGate credential store that stores login credentials.
CRYPTOENGINE	Selects which cryptographic library will the OGG processes use to provide implementation of security primitives.
MARKERTABLE	Controls the name of the masterkey that Oracle GoldenGate processes in a deployment will use to retrieve the key from the wallet.
DDLTABLE	Specifies a non-default name for the DDL history table that supports DDL synchronization for Oracle.

Table 2-1 (Cont.) All GLOBALS Parameters

Parameter	Description
ENABLE_HEARTBEAT_TABLE | DISABLE_HEARTBEAT_TABLE	Enables the processing of heartbeat tables.
ENABLEMONITORING	Enables Oracle GoldenGate Monitor to view and monitor Oracle GoldenGate instances.
EXCLUDEWILDCARDOBJECTSONLY	Includes non-wildcarded source tables when a `TABLEEXCLUDE`, `SCHEMAEXCLUDE`, or `CATALOGEXCLUDE` parameter contains a wildcard.
GGSCHEMA	Specifies the name of the schema that contains the database objects that support heartbeat tables and DDL synchronization for Oracle.
HEARTBEATTABLE	Specifies the name of the heartbeat table.
MARKERTABLE	Specifies a non-default name for the DDL marker table that supports DDL synchronization for Oracle.
MAXGROUPS	Specifies the maximum number of process groups that can run in an instance of Oracle GoldenGate.
MGRSERVNAME	Specifies the name of the Manager process when it is installed as a Windows service.
NAMECCSID	Specifies a DB2 for i CCSID if the object names in the SQL catalog are of a different CCSID than the system.
NODUPMSGSUPPRESSION	Prevents the automatic suppression of duplicate informational and warning messages.
OUTPUTFILEUMASK	Specifies a umask that can be used by Oracle GoldenGate processes to create trail files and discard files.

Table 2-1 (Cont.) All GLOBALS Parameters

Parameter	Description
TRAILBYTEORDER	Specifies the byte order (endianness) of a file created with the EXTFILE, RMTFILE, EXTTRAIL, or RMTTRAIL parameter.
TRAIL_SEQLEN_6D | TRAIL_SEQLEN_9D	Controls the sequence length for trail files.
UPREPORT	Specifies the frequency with which Manager reports Extract and Replicat processes that are running. Every time one of those processes starts or stops, events are generated.
USE_TRAILDEFS | NO_USE_TRAILDEFS	Controls where the data pump and Replicat processes obtain the table definitions when the trail files contain full table definitions.
USEIPV4	Forces Oracle GoldenGate to use IPv4 for TCP/IP connections.
USEIPV6	Forces Oracle GoldenGate to use IPv6 for TCP/IP connections.
WALLETLOCATION	Specifies the location of the master key wallet.
XAGENABLE	Enables the Oracle GoldenGate Transparent Integration with Clusterware feature that allows you to continue using GGSCI to start and stop manager when GoldenGate instance is under the management of Oracle Grid Infrastructure Bundled Agents (XAG).

2.1.2 Summary of Parameters Common to Extract and Replicat

These parameters are available for both the Extract and Replicat processes.

Parameter	Description
ALLOCFILES	Controls the incremental number of memory structures that are allocated after the initial memory allocation specified by the NUMFILES parameter is reached.

Parameter	Description
CHARSET	Specifies a multibyte character set for the process to use instead of the operating system default when reading the parameter file.
CHECKPARAMS	Verifies parameter file syntax.
COMMENT | --	Denotes comments in a parameter file.
GETENV	Retrieves variables that were set with the SETENV parameter.
OBEY	Processes parameter statements contained in a different parameter file.
SETENV	Specifies a value for a UNIX environment variable from within the GGSCI interface.
TRACETABLE | NOTRACETABLE	Specifies a trace table to which Replicat adds a record whenever it updates the target database. Causes Extract to ignore database changes generated by Replicat. Supports Oracle bi-directional replication.
USERID | NOUSERID	Specifies database connection information.
USERIDALIAS	Specifies database connection information when a credential store is in use.

These parameters are common to Extract and Replicat: Selection, Converting, and Mapping Data.

Parameter	Description
ALLOWDUPTARGETMAP | NOALLOWDUPTARGETMAP	Allows the same source-target MAP statement to appear more than once in the parameter file.
ASCIITOEBCDIC	Converts ASCII text to EBCDIC for DB2 on z/OS systems running UNIX System Services.
CATALOGEXCLUDE	Excludes the specified source container or catalog from a wildcard specification.
COLMATCH	Establishes global column-mapping rules.
DDL	Enables and filters the capture of DDL operations.
DDLSUBST	Enables string substitution in DDL processing.
GETDELETES | IGNOREDELETES	Controls the extraction of delete operations.
GETINSERTS | IGNOREINSERTS	Controls the extraction of insert operations.
GETTRUNCATES | IGNORETRUNCATES	Controls the extraction of truncate statements.

Parameter	Description
GETUPDATEAFTERS | IGNOREUPDATEAFTERS	Controls the extraction of update after images.
GETUPDATEBEFORES | IGNOREUPDATEBEFORES	Controls the extraction of update before images.
GETUPDATES | IGNOREUPDATES	Controls the extraction of update operations.
NAMECCSID	Specifies a DB2 for i CCSID if the object names in the SQL catalog are of a different CCSID than the system.
PROCEDURE	Specifies which feature groups of procedure calls are included or excluded during procedural replication.
REPLACEBADCHAR	Replaces invalid character values with another value.
SCHEMAEXCLUDE	Excludes the specified source schema from a wildcard specification.
SOURCEDEFS	Specifies a file that contains source data definitions created by the DEFGEN utility.
SOURCECATALOG	Specifies a default container or catalog for all following TABLE or MAP statements.
TRIMSPACES | NOTRIMSPACES	Controls whether trailing spaces are trimmed or not when mapping CHAR to VARCHAR columns.
VARWIDTHNCHAR | NOVARWIDTHNCHAR	Controls whether length information is written to the trail for NCHAR columns.

These parameters are common to Extract and Replicat: Custom Processing.

Parameter	Description
CUSEREXIT	Invokes a user exit routine during processing.
INCLUDE	Invokes a macro library.
MACRO	Defines an Oracle GoldenGate macro.
MACROCHAR	Defines a macro character other than the default of #.
SQLEXEC	Executes a stored procedure or query during Extract processing.
PTKCAPTUREPROCSTATS	Enables the capture of process and thread statistics for the PTK Monitoring.
PTKMONITORFREQUENCY	Sets the PTK Monitoring collection frequency interval.

These parameters common are to Extract and Replicat: Reporting.

Parameter	Description
CMDTRACE	Displays macro expansion steps in the report file.
LIST | NOLIST	Displays or suppresses the listing of macros in the report file.
REPORT	Schedules a statistical report.
STATOPTIONS	Specifies information to include in statistical displays.
REPORTCOUNT	Reports the number of records processed.
TRACE | TRACE2	Shows processing information to assist in revealing processing bottlenecks.

These parameters are common to Extract and Replicat: Tuning.

Parameter	Description
ALLOCFILES	Controls the number of incremental memory structures allocated when the value of `NUMFILES` is reached.
CACHEMGR	Manages virtual memory resources.
CHECKPOINTSECS	Controls how often the process writes a checkpoint.
DBOPTIONS	Specifies database options.
DDLOPTIONS	Specifies DDL processing options.
EOFDELAY | EOFDELAYCSECS	Determines how long the process delays before searching for more data to process in its data source.
FUNCTIONSTACKSIZE	Controls the size of the memory stack that is used for processing Oracle GoldenGate functions.
NUMFILES	Controls the initial allocation of memory dedicated to storing information about tables to be processed by Oracle GoldenGate.

These parameters are common to Extract and Replicat: Error Handling.

Parameter	Description
DDLERROR	Controls error handling for DDL extraction.
DISCARDFILE | NODISCARDFILE	Contains records that could not be processed.

These parameters are common to Extract and Replicat: Maintenance.

Parameter	Description
DISCARDROLLOVER	Controls how often to create a new discard file.
PURGEOLDEXTRACTS for Extract and Replicat	Purges obsolete trail files.
REPORTROLLOVER	Specifies when to create new report files.

2.1.3 Summary of Extract Parameters

The Extract process captures either full data records or transactional data changes, depending on configuration parameters, and then sends the data to a target system to be applied to target tables or processed further by another process, such as a load utility.

Table 2-2 Extract Parameters: General

Parameter	Description
ABORTDISCARDRECS	Controls the number of discarded records after which Extract aborts.
RECOVERYOPTIONS	Controls the recovery mode of the Extract process.
SOURCEDB	Specifies the data source as part of the login information.
TCPSOURCETIMER | NOTCPSOURCETIMER	Adjusts timestamps of records transferred to other systems when those systems reflect different times.
UPDATERECORDFORMAT	Controls whether before and after images are stored in one trail record or two.

Table 2-3 Extract Parameters: Processing Method

Parameter	Description
EXTRACT	Defines an Extract group as an online process.
GETAPPLOPS | IGNOREAPPLOPS	Controls whether or not operations from all processes except Replicat are written to a trail or file.
GETREPLICATES | IGNOREREPLICATES	Controls whether or not replicated operations are captured by an Extract on the same system.
RMTTASK	Creates a processing task on a remote system.
SOURCEISTABLE	Extracts entire records from source tables.
VAM	Indicates that a Teradata Access Module (TAM) is being used to provide transactional data to the Extract process.

Table 2-4 Extract Parameters: Selecting, Converting, and Mapping Data

Parameter	Description
COMPRESSDELETES | NOCOMPRESSDELETES	Controls whether Oracle GoldenGate writes only the key or all columns to the trail for delete operations.

Table 2-4 (Cont.) Extract Parameters: Selecting, Converting, and Mapping Data

Parameter	Description
COMPRESSUPDATES | NOCOMPRESSUPDATES	Causes only primary key columns and changed columns to be logged for updates.
EXCLUDEHIDDENCOLUMNS	The parameter disables all the Oracle Database hidden columns including the timestamp columns created using automatic CDR.
EXCLUDETAG	Specifies Replicat or data pump changes to be excluded from trail files.
FETCHOPTIONS	Controls certain aspects of the way that Oracle GoldenGate fetches data.
LOGALLSUPCOLS	Logs the columns that are required to support Conflict Detection and Resolution and Integrated Replicat.
SEQUENCE	Specifies sequences for synchronization.
TABLE | MAP	Specifies tables for extraction and controls column mapping and conversion.
TABLEEXCLUDE	Excludes source tables from the extraction process.
TARGETDEFS	Specifies a file containing target table definitions for target databases that reside on the NonStop platform.
TRAILCHARSETASCII	Specifies the ASCII character set for data captured from DB2 on z/OS, when both ASCII and EBCDIC tables are present.
TRAILCHARSETEBCDIC	Specifies the EBCDIC character set for data captured from DB2 on z/OS, when both ASCII and EBCDIC tables are present.

Table 2-5 Extract Parameters: Routing Data

Parameter	Description
EXTFILE	Specifies an extract file to which extracted data is written on the local system.
EXTTRAIL	Specifies a trail to which extracted data is written on the local system.
RMTFILE	Specifies an extract file to which extracted data is written on a remote system.
RMTHOST	Specifies the target system and Manager port number.
RMTTRAIL	Specifies a trail to which extracted data is written on a remote system.

Table 2-6 Extract Parameters: Tuning

Parameter	Description
BR	Controls the Bounded Recovery feature of Extract.
CACHEMGR	Controls the virtual memory cache manager.
FLUSHSECS | FLUSHCSECS	Determines the amount of time that record data remains buffered before being written to the trail.
LOBMEMORY	Controls the amount of memory and temporary disk space available for caching transactions that contain LOBs.
RMTHOSTOPTIONS	Specifies connection attributes other than host information for a TCP/IP connection used by a passive Extract group.
THREADOPTIONS	Controls aspects of the way that Extract operates in an Oracle Real Application Cluster environment.
TRANLOGOPTIONS	Supplies capture processing options.
TRANSMEMORY	Controls the amount of memory and temporary disk space available for caching uncommitted transaction data.
WARNLONGTRANS	Defines a long-running transaction and controls the frequency of checking for and reporting them.

Table 2-7 Extract Parameters: Maintenance

Parameter	Description
ROLLOVER	Specifies the way that trail files are aged.

Table 2-8 Extract Parameters: Security

Parameter	Description
DECRYPTTRAIL	Required to decrypt data when Extract is used as a data pump and must do work on the data.
ENCRYPTTRAIL | NOENCRYPTTRAIL	Controls encryption of data in a trail or extract file.

2.1.4 Summary of Replicat Parameters

The Replicat process reads data extracted by the Extract process and applies it to target tables or prepares it for use by another application, such as a load utility.

Parameter	Description
TARGETDB	Specifies the data source as part of the login information.

Parameter	Description
HAVEUDTWITHNCHAR	Causes Replicat to connect in UTF-8 to prevent data loss when the record being processed is a user-defined type that has an NCHAR/ NVARCHAR2 attribute.

Replicat parameters: Processing method.

Parameter	Description
BEGIN	Specifies a starting point for Replicat processing. Required when SPECIALRUN is specified.
BULKLOAD	Loads data directly into the interface of the Oracle SQL*Loader utility.
END	Specifies a stopping point for Replicat processing. Required when using SPECIALRUN.
#unique_223	Generates run and control files that are compatible with a database load utility.
REPLICAT	Specifies a Replicat group for online change synchronization.
SPECIALRUN	Used for one-time processing that does not require checkpointing from run to run.

Replicat parameters: Selecting, converting, and mapping data.

Parameter	Description
DBOPTIONS	Specifies database options. This is a global parameter, applying to all TABLE or MAP statements in the parameter file. Some DBOPTIONS options apply only to Extract or Replicat.
ALLOWNOOPUPDATES | NOALLOWNOOPUPDATES	Controls how Replicat responds to a *no-op* operation. A no-op operation is one in which there is no effect on the target table.
APPLYNOOPUPDATES | NOAPPLYNOOPUPDATES	Force a no-op update to be applied using all columns in both the SET and WHERE clauses.
APPLY_PARALLELISM	Configures number of appliers. This controls the number of connections in the target database used to apply the changes. The default value is four.
ASSUMETARGETDEFS	Assumes that source and target tables have the same column structure.
INSERTALLRECORDS	Inserts a new record into the target table for every change operation made to a record.
INSERTDELETES | NOINSERTDELETES	Converts deletes to inserts.
INSERTMISSINGUPDATES | NOINSERTMISSINGUPDATES	Converts an update to an insert when the target row does not exist.
INSERTUPDATES | NOINSERTUPDATES	Converts updates to inserts.
TABLE | MAP	Specifies a relationship between one or more source and target tables and controls column mapping and conversion.
MAPEXCLUDE	Excludes source tables from being processed by a wildcard specification supplied in MAP statements.

Parameter	Description
MAP_PARALLELISM	Configures the number of mappers. This controls the number of threads used to read the trail file. The minimum value is 1, maximum value is 100 and the default value is 2.
MAX_PARALLELISM , MIN_PARALLELISM	APPLY_PARALLELISM is auto-tuned. You can set a minimum and maximum value to define the ranges in which the Replicat automatically adjusts its parallelism. There are no defaults. Do not use with APPLY_PARALLELISM at the same time.
PRESERVETARGETTIMEZONE	Overrides the default Replicat session time zone.
REPLACEBADNUM	Specifies a global substitution value for invalid numeric data encountered when mapping number columns.
SOURCECHARSET	Controls whether the source character set it converted to the target character set.
SOURCETIMEZONE	Specifies the time zone of the source database for Replicat to use as the session time zone.
SPACESTONULL | NOSPACESTONULL	Controls whether or not a target column containing only spaces is converted to NULL.
TABLE for Replicat	Specifies a table or tables for which event actions are to take place when a row satisfies the given filter criteria.
TRAILCHARSET	Specifies the character set of the source data when the trail is of an older version that does not store the source character set, or to override the character set that is stored in the trail.
UPDATEINSERTS | NOUPDATEINSERTS	Converts insert operations to update operations for all MAP statements that are specified after it in the parameter file.
UPDATEDELETES | NOUPDATEDELETES	Converts deletes to updates.
USEDEDICATEDCOORDINATIONTHREAD	Specifies a dedicated thread for barrier transactions when Replicat is in coordinated mode.

Replicat parameters: Routing data.

Parameter	Description
EXTFILE	Defines the name of an extract file on the local system that contains data to be replicated. Used for one-time processing.
EXTTRAIL	Defines a trail containing data to be replicated. Used for one-time processing.

Replicat parameters: Error handling and reporting.

Parameter	Description
HANDLECOLLISIONS | NOHANDLECOLLISIONS	Handles errors for duplicate and missing records.

Parameter	Description
HANDLETPKUPDATE	Prevents constraint errors associated with replicating transient primary key updates.
OVERRIDEDUPS | NOOVERRIDEDUPS	Overlays a replicated insert record onto an existing target record whenever a duplicate-record error occurs.
RESTARTCOLLISIONS | NORESTARTCOLLISIONS	Controls whether or not Replicat applies HANDLECOLLISIONS logic after Oracle GoldenGate has abended because of a conflict.
REPERROR	Determines how Replicat responds to database errors.
REPFETCHEDCOLOPTIONS	Determines how Replicat responds to operations for which a fetch from the source database was required.
SHOWSYNTAX	Causes Replicat to print its SQL statements to the report file.
SQLDUPERR	Specifies the database error number that indicates a duplicate record. Use with OVERRIDEDUPS.
WARNRATE	Determines how often database errors are reported.

Replicat parameters: Tuning.

Parameter	Description
BATCHSQL	Increases the throughput of Replicat processing by arranging similar SQL statements into arrays and applying them at an accelerated rate.
COORDSTATINTERVAL	The interval at which the coordinator thread sends a request to the apply threads for statistics.
COORDTIMER	The amount of time that the coordinator thread waits for the apply threads to start.
CHUNK_SIZE	Controls how large a transaction must be for parallel Replicat to consider it as large. When parallel Replicat encounters a transaction larger than this size, it will serialize it, resulting in decreased performance. However, increasing this value will also increase the amount of memory consumed by parallel Replicat.
COMMIT_SERIALIZATION	Enables commit FULL serialization mode, which forces transactions to be committed in trail order.
DEFERAPPLYINTERVAL	Specifies a length of time for Replicat to wait before applying replicated operations to the target database.
GROUPTRANSOPS	Controls the number of records that are grouped into a Replicat transaction.
INSERTAPPEND | NOINSERTAPPEND	Controls whether or not Replicat uses an APPEND hint when applying INSERT operations to Oracle target tables.

Parameter	Description
LOOK_AHEAD_TRANSACTIONS	Controls how far ahead the Scheduler looks when batching transactions. The default value is 10000.
MAXDISCARDRECS	Limits the number of discarded records reported to the discard file.
MAXSQLSTATEMENTS	Controls the number of prepared SQL statements that can be used by Replicat.
MAXTRANSOPS	Divides large source transactions into smaller ones on the target system.
NUMFILES	Controls the initial allocation of memory that is dedicated to storing information about tables to be processed by Oracle GoldenGate.
SPLIT_TRANS_REC	Specifies that large transactions should be broken into pieces of specified size and applied in parallel. Dependencies between pieces are still honored. Disabled by default.
TRANSACTIONTIMEOUT	Specifies a time interval after which Replicat will commit its open target transaction and roll back any incomplete source transactions that it contains, saving them for when the entire source transaction is ready to be applied.

2.1.5 Summary of Wildcard Exclusion Parameters

Table 2-9 Wildcard Exclusion Parameters

Parameter	Description
EXCLUDEWILDCARDOBJECTSONLY	Forces the inclusion of non-wildcarded source objects specified in TABLE or MAP parameters when an exclusion parameter contains a wildcard that otherwise would exclude that object.
MAPEXCLUDE	Excludes a source object from a MAP statement.
TABLEEXCLUDE	Excludes a source object from a TABLE statement.
CATALOGEXCLUDE	Excludes source objects in the specified source container or catalog from the Oracle GoldenGate configuration when the container or catalog name is being specified with a wildcard in TABLE or MAP statements.
SCHEMAEXCLUDE	Excludes source objects that are owned by the specified source owner (such as a schema) from the Oracle GoldenGate configuration when wildcards are being used to specify the owners in TABLE or MAP statements.

2.1.6 Summary of DEFGEN Parameters

DEFGEN creates a file with data definitions for source or target tables. Data definitions are needed when the source and target tables have different definitions or the databases are of different types.

Table 2-10 All DEFGEN Parameters

Parameter	Description
CATALOGEXCLUDE	Excludes the specified source container or catalog from a wildcard specification.
CHARSET	Specifies a multibyte character set for the process to use instead of the operating system default when reading the parameter file.
DEFSFILE	Identifies the name of the file to which DEFGEN writes the definitions
NAMECCSID	Specifies a DB2 for i CCSID if the object names in the SQL catalog are of a different CCSID than the system.
NOCATALOG	Prevents the container or catalog name from being included in the metadata.
SCHEMAEXCLUDE	Excludes the specified source schema from a wildcard specification.
SOURCEDB	Specifies the data source as part of the login information.
TABLE for DEFGEN	Identifies a table for which you want to capture a definition.
USERIDALIAS	Specifies database connection information.

2.1.7 Summary of DDL Parameters

These parameters control Oracle GoldenGate DDL support. Other parameters may be required with DDL support, but the ones here deal specifically with the DDL feature.

Table 2-11 All DDL Parameters

Parameter	Description
DDL	Enables DDL support and filters DDL.
DDLERROR	Handles errors that occur during DDL replication.
DDLOPTIONS	Configures aspects of DDL replication other than filtering and string substitution.
DDLSUBST	Enables the substitution of strings in DDL operations.
DDLTABLE	Specifies an alternate name for the DDL history table.
GGSCHEMA	Specifies the name of the schema that contains the objects that support DDL replication.
PURGEDDLHISTORY | PURGEDDLHISTORYALT	Controls the size of the DDL history table.

Table 2-11 (Cont.) All DDL Parameters

Parameter	Description
PURGEMARKERHISTORY	Controls the size of the DDL marker table.

2.2 ABORTDISCARDRECS

Valid For

Initial Load Extract

Description

Use `ABORTDISCARDRECS` to abort Extracts configured with a `DISCARDFILE` after it has discarded *`N`* number of records.

Default

Zero (0) (Do not abort Extract and any number of discards.)

Syntax

```
ABORTDISCARDRECS
```

2.3 ALLOCFILES

Valid For

Extract and Replicat

Description

Use the `ALLOCFILES` parameter to control the incremental number of memory structures that are allocated after the initial memory allocation specified by the `NUMFILES` parameter is reached. 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. The valid range of minimum value is 1

See NUMFILESfor more information.

Default

500

Syntax

```
ALLOCFILES number
```

number
The additional number of memory structures to be allocated. Do not set ALLOCFILES to an arbitrarily high number, or memory will be consumed unnecessarily. The memory structures of Oracle GoldenGate support up to two million tables.

Example

```
ALLOCFILES 1000
```

2.4 ALLOWDUPTARGETMAP | NOALLOWDUPTARGETMAP

Valid For

Extract and Replicat

Not valid for Oracle Database when running with integrated Replicat or Parallel Replicat in Integrated mode.

Description

Use the ALLOWDUPTARGETMAP and NOALLOWDUPTARGETMAP parameters to control whether or not the following are accepted in a parameter file:

- In an Extract parameter file: duplicate TABLE parameters for the same source object if the COLMAP option is used in any of them. By default, Extract abends on duplicate TABLE statements when COLMAP is used.
- In a Replicat parameter file: duplicate MAP statements for the same source and target objects. By default, duplicate MAP statements cause Replicat to abend.

If ALLOWDUPTARGETMAP is not specified and the same source and target tables are mapped more than once, only the first MAP statement is used and the others are ignored.

Default

```
NOALLOWDUPTARGETMAP
```

Syntax

```
ALLOWDUPTARGETMAP | NOALLOWDUPTARGETMAP
```

Examples

Example 1
The following Extract parameter file is permissible with ALLOWDUPTARGETMAP enabled.

```
EXTRACT extcust
USERIDALIAS tiger1
EXTTRAIL dirdat/aa
```

```
TABLE ogg.tcustmer;
EXTTRAIL dirdat/bb
TABLE ogg.tcustmer, TARGET ogg.tcustmer, COLMAP (USEDEFAULTS, col1=id,
col2=name);
```

Example 2
The following Replicat parameter file is permissible with `ALLOWDUPTARGETMAP` enabled.

```
REPLICAT repcust
USERIDALIAS tiger1
SOURCEDEFS /ggs/dirdef/source.def
ALLOWDUPTARGETMAP
GETINSERTS
GETUPDATES
IGNOREDELETES
MAP ggs.tcustmer, TARGET ggs.tcustmer, COLMAP (USEDEFAULTS, deleted_row = 'N');
IGNOREINSERTS
IGNOREUPDATES
GETDELETES
UPDATEDELETES
MAP ggs.tcustmer, TARGET ggs.tcustmer, COLMAP (USEDEFAULTS, deleted_row = 'Y');
```

Also see About Parallel Replicat.

2.5 ALLOWINVISIBLEINDEXKEYS

Valid For

GLOBALS

Description

Use the `ALLOWINVISIBLEINDEXKEYS` parameter in the `GLOBALS` file to allow Extract and Replicat to use columns that are part of an Oracle invisible index as a unique row identifier.

> **Note:**
>
> To enable trigger-based DDL replication to use Oracle invisible indexes, set the following parameter to `TRUE` in the `params.sql script`:
>
> ```
> define allow_invisible_index_keys = 'TRUE'
> ```
>
> This functionality is automatically enabled for integrated capture and Replicat.

Default

None

Syntax

```
ALLOWINVISIBLEINDEXKEYS
```

2.6 ALLOWNULLABLEKEYS | NOALLOWNULLABLEKEYS

Valid For

GLOBALS

Description

Use NOALLOWNULLABLEKEYS to change the key selection logic so that it does not consider a nullable unique key as a viable candidate for uniquely identifying a row. When disabled, the nullable unique keys are viable candidates.

You should use care when using this parameter because it impacts the contents of the trail file and all installations must be in sync when using this parameter.

Default

ALLOWNULLABLEKEYS

Syntax

```
ALLOWNULLABLEKEYS | NOALLOWNULLABLEKEYS
```

2.7 ALLOWNONVALIDATEDKEYS

Valid For

GLOBALS

Description

Use ALLOWNONVALIDATEDKEYS to allow Extract, Replicat, and GGSCI commands to use a non-validated primary key or an invalid key as a unique identifier. This parameter overrides the key selection criteria that is used by Oracle GoldenGate. When it is enabled, Oracle GoldenGate will use NON VALIDATED and NOT VALID primary keys as a unique identifier.

A key can become invalid as the result of an object reorganization or a number of other actions, but if you know the keys are valid, ALLOWNONVALIDATEDKEYS saves the downtime of re-validating them, especially in a testing environment. However, when using ALLOWNONVALIDATEDKEYS, whether in testing or in production, you accept the risk that the target data may not be maintained accurately through replication. If a key proves to be non-valid and the table on which it is defined contains more than one record with the same key value, Oracle GoldenGate might choose the wrong target row to update.

To enable ALLOWNONVALIDATEDKEYS in a configuration where DDL replication is not active, stop all processes, add ALLOWNONVALIDATEDKEYS to the GLOBALS parameter file, and then restart the processes. To disable ALLOWNONVALIDATEDKEYS again, remove it from the GLOBALS file and then restart the processes.

To enable ALLOWNONVALIDATEDKEYS functionality in a configuration where DDL support is active, take the following steps.

1. Add the ALLOWNONVALIDATEDKEYS parameter to the GLOBALS parameter file.

2. Update the GGS_SETUP table in the DDL schema by using the following SQL.

```
UPDATE owner.GGS_SETUP SET value='1' WHERE
property='ALLOWNONVALIDATEDKEYS';
COMMIT;
```

3. Restart all Oracle GoldenGate processes including Manager. From this point on, Oracle GoldenGate selects non-validated or non-valid primary keys as a unique identifier.

To disable ALLOWNONVALIDATEDKEYS functionality when DDL support is active, take the following steps.

1. Remove ALLOWNONVALIDATEDKEYS from the GLOBALS parameter file.
2. Update the record that you added to the GGS_SETUP table to 0.
3. Restart all of the Oracle GoldenGate processes.

Default

None (Disabled)

Syntax

```
ALLOWNONVALIDATEDKEYS
```

2.8 ALLOWNOOPUPDATES | NOALLOWNOOPUPDATES

Valid For

Replicat

Description

Use ALLOWNOOPUPDATES and NOALLOWNOOPUPDATES to control how Replicat responds to a no-op operation. A no-op operation is one in which there is no effect on the target table. The following are some examples of how this can occur.

- The source table has a column that does not exist in the target table, or it has a column that was excluded from replication (with a COLSEXCEPT clause). In either case, if that source column is updated, there will be no target column name to use in the SET clause within the Replicat SQL statement.
- An update is made that sets a column to the same value as the current one. The database does not log the new value, because it did not change. However, Oracle GoldenGate captures the operation as a change record because the primary key was logged, but there is no column value for the SET clause in the Replicat SQL statement.

By default (NOALLOWNOOPUPDATES), Replicat abends with an error because these types of operations do not update the database. With ALLOWNOOPUPDATES, Replicat ignores the operation instead of abending. The statistics reported by Replicat will show that an UPDATE was made, but the database does not get updated.

You can use the internal parameter APPLYNOOPUPDATES to force the UPDATE to be applied. APPLYNOOPUPDATES overrides ALLOWNOOPUPDATES. If both are specified, Replicat applies updates for which there are key columns for the source and target tables. By default, Oracle GoldenGate abends with the following message when there is a key on the source table but no key on the target table.

```
2011-01-25 02:28:42 GGS ERROR    160 Encountered an update for target table
TELLER, which has no unique key defined. KEYCOLS can be used to define a key.
Use ALLOWNOOPUPDATES to process the update without applying it to the target
database. Use APPLYNOOPUPDATES to force the update to be applied using all
columns in both the SET and WHERE clause.
```

If ALLOWNOOPUPDATES is specified when the HANDLECOLLISIONS or INSERTMISSINGUPDATES parameter is being used, and if Oracle GoldenGate has all of the target key values, Oracle GoldenGate applies an UPDATE by using all of the columns of the table in the SET clause and the WHERE clause (invoking APPLYNOOPUPDATES behavior). This is necessary so that Oracle GoldenGate can determine whether the row is present or missing. If it is missing, Oracle GoldenGate converts the UPDATE to an INSERT.

To enable ALLOWNOOPUPDATES in a configuration where DDL replication is not active, stop all processes, add ALLOWNOOPUPDATES to the GLOBALS parameter file, and then restart the processes. To disable ALLOWNOOPUPDATES again, remove it from the GLOBALS file and then restart the processes.

To enable ALLOWNOOPUPDATES functionality in a configuration where DDL support is active, take the following steps.

1. Add the ALLOWNOOPUPDATES parameter to the GLOBALS parameter file.
2. Update the GGS_SETUP table in the DDL schema by using the following SQL.

   ```
   UPDATE owner.GGS_SETUP SET value='1' WHERE
   property='ALLOWNOOPUPDATES';
   COMMIT;
   ```

3. Restart all Oracle GoldenGate processes including Manager. From this point on, Oracle GoldenGate selects non-validated or non-valid primary keys as a unique identifier.

To disable ALLOWNOOPUPDATES functionality when DDL support is active, take the following steps.

1. Remove ALLOWNOOPUPDATES from the GLOBALS parameter file.
2. Update the record that you added to the GGS_SETUP table to 0.
3. Update the GGS_SETUP table in the DDL schema by using the following:

   ```
   UPDATE owner.GGS_SETUP SET value='1' WHERE
   property='NOALLOWNULLABLEKEYS';
   COMMIT;
   ```

4. Restart all of the Oracle GoldenGate processes.

Default

NOALLOWNOOPUPDATES (only applies if the table does not have a key)

Syntax

```
ALLOWNOOPUPDATES | NOALLOWNOOPUPDATES
```

2.9 ALLOWOUTPUTDIR

Valid For

GLOBALS

Description

Use `ALLOWOUTPUTDIR` to specify the allowed output trail directory (including its subdirectories). The specified path must exist. Symbolic links are resolved before parsing and comparison.

Default

None (A directory must be specified.)

Syntax

```
ALLOWOUTPUTDIR [relative_dir_name | absolute_dir_name]
```

relative_dir_name* | *absolute_dir_name
Specify the output trail directory name with either the relative or absolute path.

2.10 APPLYNOOPUPDATES | NOAPPLYNOOPUPDATES

Valid For

Replicat

Description

Use `APPLYNOOPUPDATES` to force a no-op `UPDATE` operation to be applied by using all of the columns in the `SET` and `WHERE` clauses. See ALLOWNOOPUPDATES | NOALLOWNOOPUPDATES for a description of *no-op*.

`APPLYNOOPUPDATES` causes Replicat to use whatever data is in the trail. If there is a primary-key `UPDATE` record, Replicat uses the before columns from the source. If there is a regular (non-key) `UPDATE`, Replicat assumes that the after value is the same as the before value (otherwise it would be a primary-key update). The preceding assumes source and target keys are identical. If they are not, you must use a `KEYCOLS` clause in the `TABLE` statement on the source.

Default

```
NOAPPLYNOOPUPDATES
```

Syntax

```
APPLYNOOPUPDATES | NOAPPLYNOPUPDATES
```

2.11 APPLY_PARALLELISM | MAX_APPLY_PARALLELISM | MIN_APPLY_PARALLELISM

Valid For

Parallel Replicat

Description

Use `APPLY_PARALLELISM` to configure the number of appliers. This controls the number of connections in the target database used to apply the changes. The default value is 4. `APPLY_PARALLELISM` is auto-tuned. You can set a minimum and maximum value to define the ranges in which the Replicat automatically adjusts its parallelism. There are no defaults. Do not use with `APPLY_PARALLELISM` at the same time.

See Basic Parameters for Parallel Replicat in *Using Oracle GoldenGate for Oracle Database*

Syntax

```
MIN_APPLY_PARALLELISM value
MAX_APPLY_PARALLELISM value
```

Example

```
MIN_APPLY_PARALLELISM 2

MAX_APPLY_PARALLELISM 10
```

2.12 ASCIITOEBCDIC

Valid For

Extract data pump and Replicat

Description

Use the `ASCIITOEBCDIC` parameter to control the conversion of data in the input trail file from ASCII to EBCDIC format. This parameter should only be used to support backward compatibility in cases where the input trail file was created by an Extract version prior to v10.0. It is ignored for all other cases, because ASCII to EBCDIC conversion is currently the default.

As of version 11.2.1, conversion is not allowed by a data pump.

Default

None

Syntax

```
ASCIITOEBCDIC
```

2.13 ASSUMETARGETDEFS

Valid For

Replicat for trail file formats prior to 12c (12.2.0.1)

Description

Use the ASSUMETARGETDEFS parameter when the source and target objects specified in a MAP statement have identical column structure, such as when synchronizing a hot site. It directs Oracle GoldenGate to assume that the data definitions of the source and target objects are identical, and to refer to the target definitions when metadata is needed for the source data.

When source and target tables have dissimilar structures, do not use ASSUMETARGETDEFS. Create a data-definitions file for the source object, and specify the definitions file with the SOURCEDEFS parameter. See SOURCEDEFS for more information. Do not use ASSUMETARGETDEFS and SOURCEDEFS in the same parameter file.

See *Administering Oracle GoldenGate* for more information about metadata and how it applies to Oracle GoldenGate.

Default

None

Syntax

```
ASSUMETARGETDEFS [OVERRIDE]
```

OVERRIDE
By default, the table definitions from the metadata records override the definitions from any ASSUMETARGETDEFS file.
Specify OVERRIDE to request Replicat to use the definitions from the target database as the definitions for the trail records.

2.14 BATCHSQL

Valid For

Replicat

Description

Use the BATCHSQL parameter to increase the performance of Replicat. BATCHSQL causes Replicat to organize similar SQL statements into arrays and apply them at an accelerated rate. In its normal mode, Replicat applies one SQL statement at a time.

BATCHSQL is valid for:

- DB2 for i (except V5R4 or i6.1)
- DB2 LUW
- DB2 on z/OS

- Oracle
- SQL Server
- Teradata

How BATCHSQL Works

In `BATCHSQL` mode, Replicat organizes similar SQL statements into batches within a memory queue, and then it applies each batch in one database operation. A batch contains SQL statements that affect the same table, operation type (insert, update, or delete), and column list. For example, each of the following is a batch:

- Inserts to table A
- Inserts to table B
- Updates to table A
- Updates to table B
- Deletes from table A
- Deletes from table B

> **Note:**
>
> Oracle GoldenGate analyzes foreign-key referential dependencies in the batches before executing them. If dependencies exist among statements that are in different batches, more than one SQL statement per batch might be required to maintain the referential integrity.

Controlling the Number of Cached Statements

The `MAXSQLSTATEMENTS` parameter controls the number of statements that are cached. See "MAXSQLSTATEMENTS" for more information. Old statements are recycled using a least-recently-used algorithm. The batches are executed based on a specified threshold (see "Managing Memory").

Usage Restrictions

SQL statements that are treated as exceptions include:

- Statements that contain `LOB` or `LONG` data.
- Statements that contain rows longer than 25k in length.
- Statements where the target table has one or more unique keys besides the primary key. Such statements cannot be processed in batches because `BATCHSQL` does not guarantee the correct ordering for non-primary keys if their values could change.
- (SQL Server) Statements where the target table has a trigger.
- Statements that cause errors.

When Replicat encounters exceptions in batch mode, it rolls back the batch operation and then tries to apply the exceptions in the following ways, always maintaining transaction integrity:

- First Replicat tries to use normal mode: one SQL statement at a time within the transaction boundaries that are set with the GROUPTRANSOPS parameter. See "GROUPTRANSOPS" for more information.
- If normal mode fails, Replicat tries to use source mode: apply the SQL within the same transaction boundaries that were used on the source.

When finished processing exceptions, Replicat resumes BATCHSQL mode.

Table 2-12 Replicat Modes Comparison

Source Transactions *(Assumes same table and column list)*	Replicat Transaction in Normal Mode	Replicat Transaction in BATCHSQL Mode	Replicat Transactions in Source Mode
Transaction 1: INSERT DELETE **Transaction2:** INSERT DELETE **Transaction 3:** INSERT DELETE	INSERT DELETE INSERT DELETE INSERT DELETE	INSERT (x3) DELETE (x3)	**Transaction 1:** INSERT DELETE **Transaction 2:** INSERT DELETE **Transaction 3:** INSERT DELETE

When to Use BATCHSQL

When Replicat is in BATCHSQL mode, smaller row changes will show a higher gain in performance than larger row changes. At 100 bytes of data per row change, BATCHSQL has been known to improve the performance of Replicat by up to 300 percent, but actual performance benefits will vary, depending on the mix of operations. At around 5,000 bytes of data per row change, the benefits of using BATCHSQL diminish.

Managing Memory

The gathering of SQL statements into batches improves efficiency but also consumes memory. To maintain optimum performance, use the following BATCHSQL options:

```
BATCHESPERQUEUE
BYTESPERQUEUE
OPSPERBATCH
OPSPERQUEUE
```

As a benchmark for setting values, assume that a batch of 1,000 SQL statements at 500 bytes each would require less than 10 megabytes of memory.

Default

Disabled (Process in normal Replicat mode)

Syntax

```
BATCHSQL
[BATCHERRORMODE | NOBATCHERRORMODE]
[BATCHESPERQUEUE n]
[BATCHTRANSOPS n]
[BYTESPERQUEUE n]
```

```
[OPSPERBATCH n]
[OPSPERQUEUE n]
[THREADS (threadID[, threadID][, ...][, thread_range[, thread_range][, ...])]
[TRACE]
```

BATCHERRORMODE | NOBATCHERRORMODE
Sets the response of Replicat to errors that occur during BATCHSQL processing mode.

BATCHERRORMODE
Causes Replicat to try to resolve errors without leaving BATCHSQL mode. It converts inserts that fail on duplicate-record errors to updates, and it ignores missing-record errors for deletes. When using BATCHERRORMODE, use the HANDLECOLLISIONS parameter to prevent Replicat from abending.

NOBATCHERRORMODE
The default, causes Replicat to disable BATCHSQL processing temporarily when there is an error, and then retry the transaction first in normal mode and then, if normal mode fails, in source mode (same transaction boundaries as on the source).

BATCHESPERQUEUE *n*
Controls the maximum number of batches that one memory queue can contain. After BATCHESPERQUEUE is reached, a target transaction is executed.

- Minimum value is 1.
- Maximum value is 1000.
- Default is 50.

BATCHTRANSOPS *n*
Controls the maximum number of batch operations that can be grouped into a transaction before requiring a commit. When BATCHTRANSOPS is reached, the operations are applied to the target.

- Minimum value is 1.
- Maximum value is 100000.
- Default is 1000 for nonintegrated Replicat (all database types) and 50 for an integrated Oracle Replicat.

BYTESPERQUEUE *n*
Sets the maximum number of bytes that one queue can contain. After BYTESPERQUEUE is reached, a target transaction is executed.

- Minimum value is 1000000 bytes (1 megabyte).
- Maximum value is 1000000000 bytes (1 gigabyte).
- Default is 2000000 bytes (20 megabytes).

OPSPERBATCH *n*
Sets the maximum number of row operations that one batch can contain. After OPSPERBATCH is reached, a target transaction is executed.

- Minimum value is 1.
- Maximum value is 100000.

- Default is 1200.

OPSPERQUEUE *n*
Sets the maximum number of row operations in all batches that one queue can contain. After OPSPERQUEUE is reached, a target transaction is executed.

- Minimum value is 1.
- Maximum value is 100000.
- Default is 1200.

THREADS (*threadID*[, *threadID*][, ...][, *thread_range*[, *thread_range*][, ...])
Valid for BATCHESPERQUEUE, BATCHTRANSOPS, and BYTESPERQUEUE. Applies these options to the specified thread or threads of a coordinated Replicat.

***threadID*[, *threadID*][, ...]**
Specifies a thread ID or a comma-delimited list of threads in the format of threadID, threadID, threadID.

[, *thread_range*[, *thread_range*][, ...]
Specifies a range of threads in the form of threadIDlow-threadIDhigh or a comma-delimted list of ranges in the format of threadIDlow-threadIDhigh, threadIDlow-threadIDhigh.

A combination of these formats is permitted, such as threadID, threadID, threadIDlow-threadIDhigh.

TRACE
Enables detailed tracing of BATCHSQL activity to the console and to the report file. Do not set tracing without the guidance of an Oracle Support analyst.

NUMTHREADS
Enables detailed tracing of BATCHSQL activity to the console and to the report file. Do not set tracing without the guidance of an Oracle Support analyst.

- Minimum value is 0.
- Maximum value is 50.

MAXTHREADQUEUEDEPTH
Enables detailed tracing of BATCHSQL activity to the console and to the report file. Do not set tracing without the guidance of an Oracle Support analyst.

- Minimum value is 0.
- Maximum value is 50.
- Default is 10.

CHECKUNIQUEKEYS
Enables detailed tracing of BATCHSQL activity to the console and to the report file. Do not set tracing without the guidance of an Oracle Support analyst.

ERRORHANDLING
Enables detailed tracing of BATCHSQL activity to the console and to the report file. Do not set tracing without the guidance of an Oracle Support analyst.

BYPASSCHECK
Enables detailed tracing of BATCHSQL activity to the console and to the report file. Do not set tracing without the guidance of an Oracle Support analyst.

Example

```
BATCHSQL BATCHESPERQUEUE 100, OPSPERBATCH 2000
```

2.15 BEGIN

Valid For

Replicat

Description

Use the BEGIN parameter to direct Replicat to start processing at the first record in the Oracle GoldenGate trail that has a timestamp greater than, or equal to, the time specified with BEGIN. All subsequent records, including records where the timestamp is less than the specified time, are processed. Use BEGIN when SPECIALRUN is specified for the same Replicat group.

Default

None

Syntax

```
BEGIN date[time]
```

date[time]
Specifies a time at which to begin processing. Valid values are a date and optional time in the format of yyyy-mm-dd[hh:mi[:ss[.ccccc]]] based on a 24-hour clock. Seconds and centiseconds are optional.

Example

```
BEGIN 2011-01-01 04:30:00
```

2.16 BLOBMEMORY

This parameter is an alias for LOBMEMORY. See "LOBMEMORY" for more information.

2.17 BR

Valid For

Extract (Oracle only)

Description

Use the BR parameter to control the Bounded Recovery (BR) feature. This feature currently supports Oracle databases.

Bounded Recovery is a component of the general Extract checkpointing facility. It guarantees an efficient recovery after Extract stops for any reason, planned or

unplanned, no matter how many open (uncommitted) transactions there were at the time that Extract stopped, nor how old they were. Bounded Recovery sets an upper boundary for the maximum amount of time that it would take for Extract to recover to the point where it stopped and then resume normal processing.

> **Caution:**
>
> Before changing this parameter from its default settings, contact Oracle Support for guidance. Most production environments will not require changes to this parameter. You can, however, specify the directory for the Bounded Recovery checkpoint files without assistance.

How Extract Recovers Open Transactions

When Extract encounters the start of a transaction in the redo log (in Oracle, this is the first executable SQL statement) it starts caching to memory all of the data that is specified to be captured for that transaction. Extract must cache a transaction even if it contains no captured data, because future operations of that transaction might contain data that is to be captured.

When Extract encounters a commit record for a transaction, it writes the entire cached transaction to the trail and clears it from memory. When Extract encounters a rollback record for a transaction, it discards the entire transaction from memory. Until Extract processes a commit or rollback, the transaction is considered *open* and its information continues to be collected.

If Extract stops before it encounters a commit or rollback record for a transaction, all of the cached information must be recovered when Extract starts again. This applies to all transactions that were open at the time that Extract stopped.

Extract performs this recovery as follows:

- If there were no open transactions when Extract stopped, the recovery begins at the current Extract read checkpoint. This is a normal recovery.
- If there were open transactions whose start points in the log were very close in time to the time when Extract stopped, Extract begins recovery by re-reading the logs from the *beginning of the oldest open transaction*. This requires Extract to do redundant work for transactions that were already written to the trail or discarded before Extract stopped, but that work is an acceptable cost given the relatively small amount of data to process. This also is considered a normal recovery.
- If there were one or more transactions that Extract qualified as *long-running open transactions*, Extract begins its recovery with a *Bounded Recovery*.

How Bounded Recovery Works

A transaction qualifies as long-running if it has been open longer than one *Bounded Recovery interval*, which is specified with the `BRINTERVAL` option of the `BR` parameter. For example, if the Bounded Recovery interval is four hours, a long-running open transaction is any transaction that started more than four hours ago.

At each Bounded Recovery interval, Extract makes a *Bounded Recovery checkpoint*, which persists the current state and data of Extract to disk, including the state and data (if any) of long-running transactions. If Extract stops after a Bounded Recovery checkpoint, it will recover from a position within the previous Bounded Recovery

interval or at the last Bounded Recovery checkpoint, instead of processing from the log position where the oldest open long-running transaction first appeared.

The *maximum Bounded Recovery time* (maximum time for Extract to recover to where it stopped) is never more than twice the current Bounded Recovery checkpoint interval. The actual recovery time will be a factor of the following:

- the time from the last valid Bounded Recovery interval to when Extract stopped.
- the utilization of Extract in that period.
- the percent of utilization for transactions that were previously written to the trail. Bounded Recovery processes these transactions much faster (by discarding them) than Extract did when it first had to perform the disk writes. This constitutes most of the reprocessing that occurs for transactional data.

When Extract recovers, it restores the persisted data and state that were saved at the last Bounded Recovery checkpoint (including that of any long running transactions).

For example, suppose a transaction has been open for 24 hours, and suppose the Bounded Recovery interval is four hours. In this case, the maximum recovery time will be no longer than eight hours worth of Extract processing time, and is likely to be less. It depends on when Extract stopped relative to the last valid Bounded Recovery checkpoint, as well as Extract activity during that time.

Advantages of Bounded Recovery

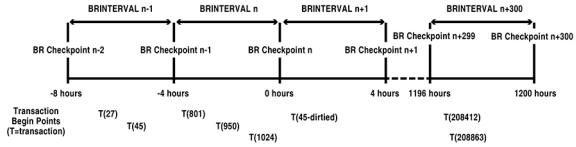
The use of disk persistence to store and then recover long-running transactions enables Extract to manage a situation that rarely arises but would otherwise significantly (adversely) affect performance if it occurred. The beginning of a long-running transaction is often very far back in time from the place in the log where Extract was processing when it stopped. A long-running transaction can span numerous old logs, some of which might no longer reside on accessible storage or might even have been deleted. Not only would it take an unacceptable amount of time to read the logs again from the start of a long-running transaction but, since long-running transactions are rare, most of that work would be the redundant capture of other transactions that were already written to the trail or discarded. Being able to restore the state and data of persisted long-running transactions eliminates that work.

Bounded Recovery Example

The following diagram illustrates a timeline over which a series of transactions were started. It shows how long-running open transactions are persisted to disk after a specific interval and then recovered after a failure. It will help to understand the terminology used in the example:

- A *persisted object* is any object in the cache that was persisted at a Bounded Recovery checkpoint. Typically this is the transactional state or data, but the cache is also used for objects that are internal to Extract. These are all collectively referred to as objects.
- The *oldest non-persisted object* is the oldest open object in the cache in the interval that immediately precedes the current Bounded Recovery checkpoint. Typically this is the oldest open transaction in that interval. Upon the restart of Bounded Recovery, runtime processing resumes from the position of the oldest non-persisted object, which, in the typical case of transactions, will be the position in the redo log of that transaction.

Figure 2-1 Sample Bounded Recovery Checkpoints



In this example, the Bounded Recovery interval is four hours. An open transaction is persisted at the current Bounded Recovery checkpoint if it has been open for more than one Bounded Recovery interval from the current Bounded Recovery checkpoint.

At BR Checkpoint n:

- There are five open transactions: T(27), T(45), T(801), T(950), T(1024). All other transactions were either committed and sent to the trail or rolled back. Transactions are shown at their start points along the timeline.
- The transactions that have been open for more than one Bounded Recovery interval are T(27) and T(45). At **BR Checkpoint n**, they are persisted to disk.
- The oldest non-persisted object is T(801). It is not eligible to be persisted to disk, because it has not been open across at least one Bounded Recovery interval. As the oldest non-persisted object, its start position in the log is stored in the **BR Checkpoint n** checkpoint file. If Extract stops unexpectedly after **BR Checkpoint n**, it will recover to that log position and start to re-read the log there. If there is no oldest non-persisted object in the preceding Bounded Recovery interval, Extract will start re-reading the log at the log position of the current Bounded Recovery checkpoint.

At BR Checkpoint n+1:

- T(45) was dirtied (updated) in the previous Bounded Recovery interval, so it gets written to a new persisted object file. The old file will be deleted after completion of **BR Checkpoint n+1**.
- If Extract fails while writing **BR Checkpoint n+1** or at any time within that Bounded Recovery checkpoint interval between **BR Checkpoint n** and **BR Checkpoint n+1**, it will recover from **BR Checkpoint n**, the last valid checkpoint. The restart position for **BR Checkpoint n** is the start of the oldest non-persisted transaction, which is T(801). Thus, the worst-case recovery time is always no more than two Bounded Recovery intervals from the point where Extract stopped, in this case no more than eight hours.

At BR Checkpoint n+3000

- The system has been running for a long time. T(27) and T(45) remain the only persisted transactions. T(801) and T(950) were committed and written to the trail sometime before **BR Checkpoint n+2999**. Now, the only open transactions are T(208412) and T(208863).
- **BR Checkpoint n+3000** is written.
- There is a power failure in the interval after **BR Checkpoint n+3000**.
- The new Extract recovers to **BR Checkpoint n+3000**. T(27) and T(45) are restored from their persistence files, which contain the state from **BR Checkpoint n**. Log reading resumes from the beginning of T(208412).

Managing Long-running Transactions

Oracle GoldenGate provides the following parameters and commands to manage long-running transactions:

- Use the `WARNLONGTRANS` parameter to specify a length of time that a transaction can be open before Extract generates a warning message that the transaction is long-running. Also use `WARNLONGTRANS` to control the frequency with which Oracle GoldenGate checks for long-running transactions. Note that this setting is independent of, and does not affect, the Bounded Recovery interval.
- Use the `SEND EXTRACT` command with the `SKIPTRANS` option to force Extract to skip a specified transaction.
- Use the `SEND EXTRACT` command with the `FORCETRANS` option to force Extract to write a specified transaction to the trail as a committed transaction.
- Use the `TRANLOGOPTIONS` parameter with the `PURGEORPHANEDTRANSACTIONS` option to enable the purging of orphaned transactions that occur when a node fails and Extract cannot capture the rollback.

About the Files that are Written to Disk

At the expiration of a Bounded Recovery interval, Extract always creates a Bounded Recovery checkpoint file. Should there be long-running transactions that require persistence, they each are written to their own persisted-object files. A persisted-object file contains the state and data of a single transaction that is persisted to disk.

Field experience has shown that the need to persist long running transactions is rare, and that the transaction is empty in most of those cases.

If a previously persisted object is still open and its state and/or data have been modified during the just-completed Bounded Recovery interval, it is re-persisted to a new persisted object file. Otherwise, previously persisted object files for open transactions are not changed.

It is theoretically possible that more than one persisted file might be required to persist a long-running transaction.

> **Note:**
>
> The Bounded Recovery files cannot be used to recover the state of Extract if moved to another system, even with the same database, if the new system is not identical to the original system in all relevant aspects. For example, checkpoint files written on an Oracle 11g Solaris platform cannot be used to recover Extract on an Oracle 11g Linux platform.

Circumstances that Change Bounded Recovery to Normal Recovery

Most of the time, Extract uses normal recovery and not Bounded Recovery, the exception being the rare circumstance when there is a persisted object or objects. Certain abnormal circumstances may prevent Extract from switching from Bounded Recovery back to normal recovery mode. These include, but are not limited to, such occurrences as physical corruption of the disk (where persisted data is stored for long-running transactions), inadvertent deletion of the Bounded Recovery checkpoint

files, and other actions or events that corrupt the continuity of the environment. There may also be more correctable reasons for failure.

In all but a very few cases, if Bounded Recovery fails during a recovery, Extract switches to normal recovery. After completing the normal recovery, Bounded Recovery is turned on again for runtime.

Bounded Recovery is not invoked under the following circumstances:

- The Extract start point is altered by CSN or by time.
- The Extract I/O checkpoint altered.
- The Extract parameter file is altered during recovery, such as making changes to the `TABLE` specification.

After completion of the recovery, Bounded Recovery will be in effect again for the next run.

What to Do if Extract Abends During Bounded Recovery

If Extract abends in Bounded Recovery, examine the error log to determine the reason. It might be something that is quickly resolved, such as an invalid parameter file or incorrect privileges on the directory that contains the Bounded Recovery files. In such cases, after the problem is fixed, Extract can be restarted with Bounded Recovery in effect.

If the problem is not correctable, attempt to restart Extract with the `BRRESET` option. Extract may recover in normal recovery mode and then turn on Bounded Recovery again.

Modifying the BR Parameter

Bounded Recovery is enabled by default with a default Bounded Recovery interval of four hours (as controlled with the `BRINTERVAL` option). This interval should be appropriate for most environments. Do not alter the `BR` parameter without first obtaining guidance from an Oracle support analyst. Bounded Recovery runtime statistics are available to help Oracle GoldenGate analysts evaluate the Bounded Recovery usage profile to determine the proper setting for `BRINTERVAL` in the unlikely event that the default is not sufficient.

Should you be requested to alter `BR`, be aware that the Bounded Recovery interval is a multiple of the regular Extract checkpoint interval. The Extract checkpoint is controlled by the `CHECKPOINTSECS` parameter. Thus, the `BR` parameter controls the ratio of the Bounded Recovery checkpoint to the regular Extract checkpoint. You might need to change both parameters, if so advised by your Oracle representative.

Supported Databases

This parameter applies to Oracle databases. For other databases, Extract recovers by reading the old logs from the start point of the oldest open transaction at the point of failure and does not persist long-running transactions.

Default

```
BR BRINTERVAL 4, BRDIR BR
```

Syntax

```
BR
[, BRDIR directory]
[, BRINTERVAL number {M | H}]
[, BRKEEPSTALEFILES]
[, BROFF]
[, BROFFONFAILURE]
[, BRFSOPTION { MS_SYNC | MS_ASYNC }]
```

BRDIR *directory*
Specifies the relative or full path name of the parent directory that will contain the BR directory. The BR directory contains the Bounded Recovery checkpoint files, and the name of this directory cannot be changed. The default parent directory for the BR directory is a directory named BR in the root directory that contains the Oracle GoldenGate installation files.
Each Extract group within a given Oracle GoldenGate installation will have its own sub-directory under the directory that is specified with BRDIR. Each of those directories is named for the associated Extract group.
For *directory*, do not use any name that contains the string temp or tmp (case-independent). Temporary directories are subject to removal during internal or external cleanup procedures.

BRINTERVAL *number* {M | H}
Specifies the time between Bounded Recovery checkpoints. This is known as the *Bounded Recovery interval*. This interval is an integral multiple of the standard Extract checkpoint interval, as controlled by the CHECKPOINTSECS parameter. However, it need not be set exactly. Bounded Recovery will adjust any legal BRINTERVAL parameter internally as it requires.
The minimum interval is 20 minutes. The maximum is 96 hours. The default interval is 4 hours.

BRKEEPSTALEFILES
Causes old Bounded Recovery checkpoint files to be retained. By default, only current checkpoint files are retained. Extract cannot recover from old Bounded Recovery checkpoint files. Retain old files only at the request of an Oracle support analyst.

BROFF
Turns off Bounded Recovery for the run and for recovery. Consult Oracle before using this option. In most circumstances, when there is a problem with Bounded Recovery, it turns itself off.

BROFFONFAILURE
Disables Bounded Recovery after an error. By default, if Extract encounters an error during Bounded Recovery processing, it reverts to normal recovery, but then enables Bounded Recovery again after recovery completes. BROFFONFAILURE turns Bounded Recovery off for the runtime processing.

BRRESET
BRRESET is a start up option that forces Extract to use normal recovery for the current run, and then turn Bounded Recovery back on after the recovery is complete. Its purpose is for the rare cases when Bounded Recovery does not revert to normal recovery if it encounters an error. Bounded Recovery is enabled during runtime. Consult Oracle Support before using this option.

To use this option, you must start Extract from the command line. To run Extract from the command line, use the following syntax:

```
extract paramfile name.prm reportfile name.rpt
```

Where:

- `paramfile name.prm` is the relative or fully qualified name of the Extract parameter file. The command name can be abbreviated to `pf`.
- `reportfile name.rpt` is the relative or fully qualified name of the Extract report file, if you want it in a place other than the default. The command name can be abbreviated to `rf`.

`BR BRFSOPTION {MS_SYNC | MS_ASYNC}`
Performs synchronous/asynchronous writes of the mapped data in Bounded Recovery.

`MS_SYNC`
Bounded Recovery writes of mapped data are synchronized for I/O data integrity completion.

`MS_ASYNC`
Bounded Recovery writes of mapped data are initiated or queued for servicing.

Example

`BR BRDIR /user/checkpt/br` specifies that the Bounded Recovery checkpoint files will be created in the `/user/checkpt/br` directory.

2.18 BULKLOAD

Valid For

Replicat

Description

Use the `BULKLOAD` parameter for an initial load Replicat when using the direct bulk load to Oracle SQL*Loader method. This method passes initial-load data directly to the interface of Oracle's SQL*Loader utility to perform a direct load. A Collector process and trails are not used.

For a complete guide to the methods of loading data with Oracle GoldenGate, see *Administering Oracle GoldenGate*.

Default

None

Syntax

```
BULKLOAD
[LOGGING | NOLOGGING]
[PARALLEL | NOPARALLEL]
[SKIPALLINDEXES | SKIPUNUSEDINDEX | NOSKIPALLINDEXES]
```

LOGGING | NOLOGGING
Valid for Replicat for Oracle. LOGGING is the default and enables redo logging for the loaded objects. NOLOGGING increases BULKLOAD performance by disabling redo logging of the loaded objects. Do not specify NOLOGGING for cascading synchronization and multiple master configurations.
However, BULKLOAD must be set to LOGGING if the target is part of a cascading or bi-directional configuration, where a local Extract will capture the loaded objects.

PARALLEL | NOPARALLEL
Valid for Replicat for Oracle. PARALLEL enables BULKLOAD to use multiple load sessions to load the same segment concurrently. NOPARALLEL is the default and disables parallel loading.

SKIPALLINDEXES | SKIPUNUSEDINDEX | NOSKIPALLINDEXES
Valid for Replicat for Oracle. Controls the handling of indexes. NOSKIPALLINDEXES is the default, which allows index maintenance during a direct load. SKIPALLINDEXES skips all index maintenance. SKIPUNUSEDINDEX skips unusable indexes.

2.19 CACHEMGR

Valid For

Extract and Replicat, all databases except DB2 on z/OS. .

Description

Use the CACHEMGR parameter to specify a non-default file system location for the temporary files needed to hold uncommitted transaction data. The CACHEMGR parameter can also be used to control the amount of virtual memory and temporary disk space that is available for caching uncommitted transaction data. Both of these latter uses are discouraged.

> **Caution:**
>
> Do not change this parameter without consulting Oracle Support. CACHEMGR is internally self-configuring and self-adjusting. It is rare that this parameter requires modification. Doing so unnecessarily may result in performance degradation. It is best to acquire empirical evidence before opening an Oracle Service Request and consulting with Oracle Support.
> However, you can specify the directory for the temporary files without assistance

Oracle GoldenGate only replicates committed transactions. Until a COMMIT is received, any transactional data is stored in an area of virtual memory known as a **cache**. This cache is managed by the CACHEMGR. If the amount of transaction data becomes too great for the virtual memory, then the CACHEMGR writes some of the cached data to temporary files on disk.

Your systems should have sufficient operating system swap and page file space. Oracle recommends a minimum of 512GB.

Identifying the Paging Directory

By default, Oracle GoldenGate maintains the transaction data that it swaps to disk a sub-directory of the Oracle GoldenGate installation directory. `CACHEMGR` assumes that all of the free space on the file system is available. This directory may fill up quickly if there is a large transaction volume with large transaction sizes. To prevent I/O contention and possible disk-related failures, dedicate a disk to this directory. You can assign directory location with the `CACHEDIRECTORY` option of the `CACHEMGR` parameter. A size can also be assigned. However, this is discouraged and should only be done after consulting Oracle Support.

Guidelines for Using CACHEMGR

- This parameter is valid for all databases supported by Oracle GoldenGate except DB2 z/OS.
- At least one argument must be supplied. `CACHEMGR` by itself is invalid.
- Parameter options can be listed in any order.
- Only one `CACHEMGR` parameter is permitted in a parameter file.

Default

None

Syntax

```
CACHEMGR {
[CACHEDIRECTORY path [size] [, CACHEDIRECTORY path [size] [, ...],]
CACHESIZE size
}
```

`CACHEDIRECTORY path [size]`
Specifies the name of the directory to which Oracle GoldenGate writes transaction data to disk temporarily when necessary. The default without this parameter is the `dirtmp` sub-directory of the Oracle GoldenGate installation directory. Any directory for temporary files can be on an Oracle Database file system, but cannot be on a direct I/O or concurrent I/O mounted file system that does not support the `mmap()` or `MapViewOfFile()` system calls, like AIX.

- *`path`* is a fully qualified directory name.
- *`size`* sets a maximum amount of disk space that can be allocated to the specified directory. The upper limit is imposed by the file system, such as the maximum file size or number of files. The minimum size is 2 GB, which is enforced. There is no default. Oracle discourages the use of the `size` option and you should only it when in consultation with Oracle Support.

You can specify more than one directory by using a `CACHEDIRECTORY` clause for each one. The maximum number of directories is 100.
The value can be specified in bytes or in terms of gigabytes, megabytes, or kilobytes in any of the following forms:
`GB | MB | KB | G | M | K | gb | mb | kb | g | m | k`

CACHESIZE *size*
Sets a soft limit for the amount of virtual memory (**CACHESIZW**) that is available for caching transaction data. You can internally adjust the CACHESIZE using CACHEMGR as necessary.
If you feel that the default CACHEMGR configuration and internal self-adjustment is adversely affecting your system performance, then you should open a Service Request with Oracle Support. It is best to have acquired empirical data showing the problem symptoms in question to aid in configuring a new default.

Example

```
CACHEMGR CACHEDIRECTORY /net/d4atd/ggs/temp
```

2.20 CATALOGEXCLUDE

Valid For

Extract, Replicat, DEFGEN

Description

Use the CATALOGEXCLUDE parameter to explicitly exclude source objects in the specified container or catalog from the Oracle GoldenGate configuration when the container or catalog name is being specified with a wildcard in TABLE or MAP statements. This parameter is valid when the database is an Oracle container databasef, where fully qualified three-part names are being used.

The positioning of CATALOGEXCLUDE in relation to parameters that specify files or trails determines its effect. Parameters that specify trails or files are: EXTFILE, RMTFILE, EXTTRAIL, RMTTRAIL. The parameter works as follows:

- When a CATALOGEXCLUDE specification is placed before any TABLE or SEQUENCE parameters, and also before the parameters that specify trails or files, it applies globally to all trails or files, and to all TABLE and SEQUENCE parameters.
- When a CATALOGEXCLUDE specification is placed after a parameter that specifies a trail or file, it is effective only for that trail or file and only for the TABLE or SEQUENCE parameters that are associated with it. Multiple trail or file specifications can be made in a parameter file, each followed by a set of TABLE, SEQUENCE, and CATALOGEXCLUDE specifications.

CATALOGEXCLUDE is evaluated before evaluating the associated TABLE or SEQUENCE parameter. Thus, the order in which they appear does not make a difference.

See also the EXCLUDEWILDCARDOBJECTSONLY parameter.

Default

None

Syntax

```
CATALOGEXCLUDE {container}
```

container
The source Oracle container that is to be excluded. A wildcard can be used. Follow the rules for using wildcards described in *Administering Oracle GoldenGate*.

Examples

Example 1
This example omits the pdb1 pluggable database.

```
EXTRACT capt
USERIDALIAS alias1
RMTHOST sysb, MGRPORT 7809
RMTTRAIL /ggs/dirdat/aa
CATALOGEXCLUDE pdb1
TABLE *.*.*;
```

2.21 CHARMAP

Valid For

Replicat

Description

Use the CHARMAP parameter to specify that the character mapping file overrides the character code point mapping. By enabling character set conversion for same character sets, you may encounter performance degradation.

Default

The encoding of the parameter file is operating system default character set.

Syntax

CHARMAP *charmap filename*

The character mapping file format is as follows:

```
       -- Sample character mapping file.
       --    Can use -- or COMMENT as comment line.
       --    Can use CHARSET parameter to specify file encoding.
       --
       -- Source character set
       SOURCECHARSET     shiftjis
       --
       -- Target character set
       TARGETCHARSET     ja16euc
       --
       -- Character map definition by one code point.
       --   left hand is source and right hand target code point.
       \xa2c1     \x89\xa2\xb7        -- override \xa2c1 to \x89\xa2\xb7
       --
       -- Character map definition by range. Number of source and target
characters must be the same.
      \x61 - \x7a        \x41 - \x5a
```

Example

In the following example, the character map definition is given using a character mapping text file:

CHARMAP *charmapdesc.txt*

```
REPLACEBADCHAR FORCECHECK
```

This enables strict character set conversion and check code point even if the source and target are the same.

Add the following to your character mapping file:

```
SOURCECHARSET   windows-932
TARGETCHARSET   windows-932
\x61 - \x7a    \x41 - \x5a
```

2.22 CHECKPARAMS

Valid For

Extract and Replicat

Description

Use the CHECKPARAMS parameter to test the syntax of a parameter file. To start the test:

1. Edit the parameter file to add CHECKPARAMS.
2. (Optional) To verify the tables, add the NODYNAMICRESOLUTION parameter.
3. Start the process. Without processing data, Oracle GoldenGate audits the syntax. If NODYNAMICRESOLUTION exists, Oracle GoldenGate connects to the database to verify that the tables specified with TABLE or MAP exist. If there is a syntax failure, the process abends with error 190. If the syntax succeeds, the process stops and writes a message to the report file that the parameters processed successfully.
4. Do one of the following:
   - If the test succeeds, edit the file to remove the CHECKPARAMS parameter and the NODYNAMICRESOLUTION parameter, if used, and then start the process again to begin processing.
   - If the test fails, edit the parameter file to fix the syntax based on the report's findings, and then remove NODYNAMICRESOLUTION and start the process again.

CHECKPARAMS can be positioned anywhere within the parameter file.

Default

None

Syntax

```
CHECKPARAMS
```

2.23 CHECKPOINTSECS

Valid For

Extract and Replicat

Description

Use the CHECKPOINTSECS parameter to control how often Extract and Replicat make their routine checkpoints.

- Decreasing the value causes more frequent checkpoints. This reduces the amount of data that must be reprocessed if the process fails, but it could cause performance degradation because data is written to disk more frequently.
- Increasing the value causes less frequent checkpoints. This might improve performance, but it increases the amount of data that must be reprocessed if the process fails. When using less frequent Extract checkpoints, make certain that the transaction logs remain available in case the data has to be reprocessed.

> **Note:**
>
> In addition to its routine checkpoints, Replicat also makes a checkpoint when it commits a transaction.

Avoid changing CHECKPOINTSECS unless you first open an Oracle service request.

Default

10 seconds

Syntax

```
CHECKPOINTSECS seconds
```

seconds
The number of seconds to wait before issuing a checkpoint.

Example

```
CHECKPOINTSECS 20
```

2.24 CHECKPOINTTABLE

Valid For

GLOBALS

Description

Use the CHECKPOINTTABLE parameter in a GLOBALS parameter file to specify the name of a default checkpoint table that can be used by all Replicat groups in one or more Oracle GoldenGate instances. All Replicat groups created with the ADD REPLICAT command will default to this table unless it is overridden by using the CHECKPOINTTABLE option of that command.

To create the checkpoint table, use the ADD CHECKPOINTTABLE command in GGSCI. Do not use a checkpoint table for a Replicat that is configured in integrated mode against an Oracle target database.

See *Administering Oracle GoldenGate* for more information about creating a checkpoint table.

Default

None

Syntax

```
CHECKPOINTTABLE [container.] owner.table
```

[container.]owner.table
The owner and name of the checkpoint table. Additionally, for an Oracle container database, specify the correct pluggable database (container).

Example

```
CHECKPOINTTABLE finance.ggs.chkpt
```

2.25 CHUNK_SIZE

Valid For

Parallel Replicat

Description

Controls how large a transaction must be for parallel Replicat to consider it as large. When parallel Replicat encounters a transaction larger than this size, it will serialize it, resulting in decreased performance. However, increasing this value will also increase the amount of memory consumed by parallel Replicat.

See Creating a Parallel Replication in *Using Oracle GoldenGate for Oracle Database*.

Default

None

2.26 CMDTRACE

Valid For

Extract and Replicat

Description

Use the `CMDTRACE` parameter to display macro expansion steps in the report file. You can use this parameter more than once in the parameter file to set different options for different macros.

Default

```
OFF
```

Syntax

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

ON
Enables the display of macro expansion.

OFF
Disables the display of macro expansion.

DETAIL
Produces a verbose display of macro expansion.

Example

In the following example, tracing is enabled before `#testmac` is invoked, and then disabled after the macro's execution.

```
MACRO #testmac
BEGIN
col1 = col2,
col3 = col4
END;
...
CMDTRACE ON
MAP test.table2 , TARGET test.table2,
COLMAP (#testmac);
CMDTRACE OFF
```

2.27 COLCHARSET

Valid For

Extract, Replicat, and DEFGEN

Description

Use `COLCHARSET` clause to specify particular column character set or disable character set conversion. This parameter overrides the column character set for the specified column.

The character set specified by the `COLCHARSET` parameter overrides the character set in the trail file, the character set specified by the `SOURCECHARSET OVERRIDE` parameter and the character set specified by the `CHARSET` parameter.

The character set specified by the `COLCHARSET` Replicat parameter overrides the column level character set specified in the source table definition file.

If the `COLCHARSET` is specified for DEFGEN file format less than level four, the parameter is ignored and warning message is issued. The column level character set attribute for the older table definition file format is not output.

The `COLCHARSET` parameter overrides the source column level character set and change the Replicat character set conversion behavior by assuming the source column character set as specified character set.

Default

None

Syntax

```
COLCHARSET character_set (column [, ...])
```

character_set
Any supported character set.

column
The name of a column. To specify multiple columns, create a comma-delimited list.

Examples

Example 1
The following example specifies multiple columns.

```
TABLE SchemaName.TableName, COLCHARSET( WE8MSWIN1252, col0, col2 );
```

Example 2
The following example specifies a different character set.

```
MAP SchemaName.*, TargetName *.*,
  COLCHARSET( WE8MSWIN1252, col1 ),
  COLCHARSET( WE8ISO8859P1, col2 )
```

Example 3
The following example specifies different character set.

```
MAP SchemaName.*, TargetName *.*,
  COLCHARSET( WE8MSWIN1252, col1 ),
  COLCHARSET( WE8ISO8859P1, col2 )
```

Example 4
The following example specifies a wildcard.

```
MAP SchemaName.*, TargetName *.*, COLCHARSET( WE8MSWIN1252, col* )
```

Example 5
The following example disables character set conversion on particular column.

```
MAP SchemaName.*, TargetName *.*, COLCHARSET(PASSTHRU, col )
```

2.28 COLMATCH

Valid For

Extract and Replicat

Description

Use the COLMATCH parameter to create global rules for column mapping. COLMATCH rules apply to all TABLE or MAP statements that follow the COLMATCH statement. Global rules can be turned off for subsequent TABLE or MAP entries with the RESET option.

With COLMATCH, you can map between tables that are similar in structure but have different column names for the same sets of data. COLMATCH provides a more convenient way to map columns of this type than does using a COLMAP clause in individual TABLE or MAP statements.

With COLMATCH, you can:

- Map explicitly based on column names.
- Ignore name prefixes or suffixes.

Either COLMATCH or a COLMAP clause of a TABLE or MAP statement is required when mapping differently named source and target columns.

See *Administering Oracle GoldenGate* for more information about mapping columns.

Default

None

Syntax

```
COLMATCH
{NAMES target_column = source_column |
PREFIX prefix |
SUFFIX suffix |
RESET}
```

NAMES *target_column = source_column*
Specifies the name of a target and source column, for example CUSTOMER_CODE and CUST_CODE. If the database requires double quotes to enforce case-sensitivity, specify the column name that way. For example: NAMES "ABC" = "ABC2". For other case-sensitive databases, specify the column name as it is stored in the database, for example: NAMES ABC = abc.

PREFIX *prefix* | SUFFIX *suffix*
Specifies a column name prefix or suffix to ignore. If the database requires double quotes to enforce case-sensitivity, specify the prefix or suffix that way if it is case-sensitive. For other case-sensitive databases, specify the prefix or suffix as it is stored in the database
For example, to map a target column named "ORDER_ID" to a source column named "P_ORDER_ID", specify:

```
COLMATCH PREFIX "P_"
```

To map a target column named "CUST_CODE_K" to a source column named CUST_CODE, specify:

```
COLMATCH SUFFIX "_K"
```

RESET
Turns off previously defined COLMATCH rules for subsequent TABLE or MAP statements.

Examples

Example 1

```
COLMATCH NAMES "CUSTOMER_CODE" = "CUST_CODE"
```

Example 2

```
COLMATCH NAMES Customer_Code = "Cust_Code"
```

Example 3

```
COLMATCH PREFIX P_
```

Example 4

```
COLMATCH SUFFIX _K
```

Example 5

```
COLMATCH RESET
```

2.29 COMPRESSDELETES | NOCOMPRESSDELETES

Valid For

Extract

Description

Use the COMPRESSDELETES and NOCOMPRESSDELETES parameters to control the way that columns are written to the trail record for DELETE operations.

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.

These parameters support the following databases:

DB2 LUW
DB2 z/OS
DB2 for i
MySQL
SQL Server
Teradata

Default

```
COMPRESSDELETES
```

Syntax

```
{COMPRESSDELETES | NOCOMPRESSDELETES [FETCHMISSINGCOLUMNS]}
```

COMPRESSDELETES
Causes Extract to write only the primary key to the trail for DELETE operations. This is the default. The key provides enough information to delete the correct target record, while restricting the amount of data that must be processed.

NOCOMPRESSDELETES [FETCHMISSINGCOLUMNS]
NOCOMPRESSDELETES sends all of the columns to the trail. This becomes the default when a table definition does not include a primary key or unique index, or when a substitute key is defined with the KEYCOLS option of TABLE. The KEYCOLS option writes the specified columns to the trail whether or not a real key exists. See KEYCOLS (*columns*) for more information about the KEYCOLS option.
NOCOMPRESSDELETES is also required when using the Conflict Detection and Resolution (CDR) feature for a DB2 database on any of the platforms that are supported by Oracle GoldenGate. See *Administering Oracle GoldenGate* for more information about CDR.

FETCHMISSINGCOLUMNS is valid for Oracle Database only. It causes the values of data types that are only supported by fetching to be fetched from the database on DELETE operations. These data types are LOB, UDT, LONG, and some XMLType columns. For detailed information about columns that are supported by fetching (rather than directly captured from the redo stream), see Configuring a Downstream Mining Database in *Using Oracle GoldenGate for Oracle Database*. The columns that are fetched will appear in the trail file as part of the DELETE record. If NOCOMPRESSDELETES is used for Oracle Database data without the FETCHMISSINGCOLUMNS option, only the LOB data that can be read from the logs (without fetching) will be included in the DELETE operation in the trail.

2.30 COMPRESSUPDATES | NOCOMPRESSUPDATES

Valid For

Extract

Description

Use the COMPRESSUPDATES and NOCOMPRESSUPDATES parameters for Extract to control the way columns are written to the trail record for UPDATE operations.

COMPRESSUPDATES, the default, causes Extract to write only the primary key and the changed columns of a row to the trail for update operations. This provides enough information to update the correct target record (unless conflict resolution is required), while restricting the amount of data that must be processed.

If other columns are supplementally logged by the database for use by Oracle GoldenGate, Extract writes those columns to the trail as well. Additionally, if a substitute key is defined with the KEYCOLS option of the TABLE parameter, those columns are written to the trail, whether or not a primary or unique key is defined. See "KEYCOLS (*columns*)" for more information.

NOCOMPRESSUPDATES sends all of the columns to the trail. This becomes the default when a table definition does not include a primary key or unique index. NOCOMPRESSUPDATES also is required when using the Conflict Detection and Resolution (CDR) feature for a DB2 database on any of the platforms that are supported by Oracle GoldenGate. See *Administering Oracle GoldenGate* for more information about CDR.

COMPRESSUPDATES and NOCOMPRESSUPDATES apply globally for all TABLE statements in a parameter file.

These parameters support the following databases:

DB2 LUW
DB2 z/OS
DB2 for i
MySQL
Oracle
SQL Server
Teradata

Default

`COMPRESSUPDATES`

Syntax

`COMPRESSUPDATES | NOCOMPRESSUPDATES`

2.32 COORDSTATINTERVAL

Valid For

Replicat in coordinated mode

Description

Use the `COORDSTATINTERVAL` parameter to set the amount of time, in seconds, between requests for statistics sent by the Replicat coordinator thread to the apply threads. If a thread does not return statistics within an internal heartbeat interval, Replicat logs a warning message. The heartbeat interval is not configurable and is always six times the `COORDSTATINTERVAL` interval. At the default `COORDSTATINTERVAL` interval of 10 seconds, for example, the heartbeat default is one minute (60 seconds).

Default

The minimum value is 0; the maximum value is 2147483647. The default value is 10 seconds

Syntax

`COORDSTATINTERVAL` *`interval`*

`interval`
The interval, in seconds, between requests for thread statistics. Valid values are 0 or any positive number.

2.33 COORDTIMER

Valid For

Replicat in coordinated mode

Description

Use the `COORDTIMER` parameter to set a base amount of time, in seconds, that the threads and coordinator wait for each other to start. A thread will wait for this base time interval before retrying a connection to the coordinator and it will do this a certain number of times. The coordinator waits for the length of this base time interval and it is reset after every thread is successfully registered. The overall time the coordinator waits before abending is dependent on this timer and it is variable depending on the register time of the threads.

A value of 0 disables this timing procedure. If timing is disabled, the coordinator thread may wait indefinitely for the threads to start, and Replicat will enter a suspended

state. In this case, the internal Replicat heartbeat timer is disabled regardless of the COORDSTATINTERVAL setting.

Default

The minimum value is 0; the maximum value is 2147483647. The default value is 180 seconds (three minutes)

Syntax

```
COORDTIMER wait_time
```

wait_time
The amount of time, in seconds, that the coordinator thread waits for the apply threads to start. Valid values are 0 or any positive number.

2.34 CREDENTIALSTORELOCATION

Valid For

GLOBALS

Description

Use the `CREDENTIALSTORE` parameter to change the location of the Oracle GoldenGate credential store from the default location. The default location is the `dircrd` directory in the Oracle GoldenGate installation directory. The default location is the preferred location.

The credential store stores database user names and passwords in encrypted format as a security measure. When `CREDENTIALSTORE` is used, the specified location is assumed for all GGSCI commands that manage the credential store.

Syntax

```
CREDENTIALSTORELOCATION directory
```

directory
The full path name of the directory where the credential store is to be stored.

Example

```
CREDENTIALSTORELOCATION /home/ogg/credentials
```

2.35 CRYPTOENGINE

Valid For

GLOBALS

Description

Use the `CRYPTOENGINE` to select which cryptographic library the Oracle GoldenGate processes use to provide implementation of security primitives.

Syntax

```
CRYPTOENGINE (CLASSIC | FIPS140 | NATIVE)
```

CRYPTOENGINE
Selects which cryptographic library will the OGG processes use to provide implementation of security primitives.

CLASSIC
Uses the Oracle NNZ security framework without FIPS-140 enhancements.

FIPS140
Uses the Oracle NNA security framework, but enhanced with the FIPS-140.2 compliant version of the RSA MES shared libraries.

NATIVE
For the platforms where this is available, it will use a native library that makes more efficient use of the CPU cryptographic primitives, resulting in higher product throughput when using trail and TCP encryption. Currently, Intel's IPP library version 9.0 is used for Linux.x64 and Windows.x64. All other platforms fall back to `CLASSIC` behavior.

2.36 CUSEREXIT

Valid For

Extract when fetching from a multitenant container database (CDB) and Replicat

Description

Use the `CUSEREXIT` parameter to call a custom exit routine written in C programming code from a Windows DLL or UNIX shared object at a defined exit point within Oracle GoldenGate processing. Your user exit routine must be able to accept different events and information from the Extract and Replicat processes, process the information as desired, and then return a response and information to the caller (the Oracle GoldenGate process that called it).

User exits can be used as an alternative to, or in conjunction with, the data transformation functions that are available within the Oracle GoldenGate solution.

> **Note:**
>
> When using a coordinated Replicat to call a user exit routine, you are responsible for writing the user exits in a thread-safe manner.

For help with creating and implementing user exits, see *Administering Oracle GoldenGate*.

Default

None

Syntax

```
CUSEREXIT {DLL | shared_object} routine
[, INCLUDEUPDATEBEFORES]
[, PARAMS 'string']
```

{DLL | shared_object}
The name of the Windows DLL or UNIX shared object that contains the user exit function.

routine
The name of the exit routine to be executed.

INCLUDEUPDATEBEFORES
Passes the before images of column values to a user exit. When using this parameter, you must explicitly request the before image by setting the `requesting_before_after_ind` flag to `BEFORE_IMAGE_VAL` within a callback function that supports this flag. Otherwise, only the after image is passed to the user exit. By default, Oracle GoldenGate only works with after images.
When using `INCLUDEUPDATEBEFORES` for a user exit that is called from a data pump or from Replicat, always use the `GETUPDATEBEFORES` parameter for the primary Extract process, so that the before image is captured, written to the trail, and causes a `process_record` event in the user exit. In a case where the primary Extract also has a user exit, `GETUPDATEBEFORES` causes both the before image and the after image to be sent to the user exit as separate `EXIT_CALL_PROCESS_RECORD` events.
If the user exit is called from a primary Extract (one that reads the transaction log), only `INCLUDEUPDATEBEFORES` is needed for that Extract. `GETUPDATEBEFORES` is not needed in this case, unless other Oracle GoldenGate processes downstream will need the before image to be written to the trail. `INCLUDEUPDATEBEFORES` does not cause before images to be written to the trail.

PARAMS *'string'*
Passes the specified string at startup. Can be used to pass a properties file, startup parameters, or other string. Enclose the string within single quote marks.
Data in the string is passed to the user exit in the `EXIT_CALL_START` `exit_params_def.function_param`. If no quoted string is specified with `PARAMS`, the `exit_params_def.function_param` is `NULL`.

Examples

Example 1

```
CUSEREXIT userexit.dll MyUserExit
```

Example 2

```
CUSEREXIT userexit.dll MyUserExit, PARAMS 'init.properties'
```

Example 3

```
CUSEREXIT userexit.dll MyUserExit, INCLUDEUPDATEBEFORES, PARAMS
'init.properties'
```

Example 4

```
CUSEREXIT userexit.dll MyUserExit, INCLUDEUPDATEBEFORES, &
  PARAMS 'init.properties'
```

Example 5

```
CUSEREXIT cuserexit.dll MyUserExit, &
  INCLUDEUPDATEBEFORES, PARAMS 'Some text to start with during startup'
```

2.37 DBOPTIONS

Valid For

Extract and Replicat

Description

Use the DBOPTIONS parameter to specify database options. This is a global parameter, applying to all TABLE or MAP statements in the parameter file. Some DBOPTIONS options apply only to Extract or Replicat.

The DBOPTIONS parameter can be placed anywhere in the parameter file irrespective of other parameters.

Default

None

Syntax

```
DBOPTIONS
[ALLOWLOBDATATRUNCATE | NOALLOWLOBDATATRUNCATE]
[ALLOWUNUSEDCOLUMN | NOALLOWUNUSEDCOLUMN]
[BINDCHARFORBITASCHAR]
[CATALOGCONNECT | NOCATALOGCONNECT]
[CONNECTIONPORT port]
[DECRYPTPASSWORD shared_secret ENCRYPTKEY {DEFAULT | key_name}]
[DEFERREFCONST]
[DISABLECOMMITNOWAIT]
[DISABLELOBCACHING]
[ENABLE_INSTANTIATION_FILTERING]
[EMPTYLOBSTRING 'string']
[FETCHBATCHSIZE records]
[FETCHCHECKFREQ seconds]
[FETCHLOBS | NOFETCHLOBS]
[FETCHRETRYCOUNT number]
[FECHTIMEOUT seconds | NOFECHTIMEOUT]
[HOST {DNS_name | IP_address}]
[INTEGRATEDPARAMS(parameter[, ...])]
[LIMITROWS | NOLIMITROWS]
[LOBBUFSIZE bytes]
[LOBWRITESIZE bytes]
[SESSIONPOOLMAX max_value |
[SESSIONPOOLMIN min_value][SESSIONPOOLINCR increment_value]
[SETTAG [tag_value | NULL] ]
[SHOWINFOMESSAGES]
[SHOWWARNINGS]
[SKIPTEMPLOB | NOSKIPTEMPLOB]
[SOURCE_DB_NAME src_dbase_global_name]
[SPTHREAD | NOSPTHREAD]
[SQLMODE]
[SUPPRESSTEMPORALUPDATES]
[SUPPRESSTRIGGERS | NOSUPPRESSTRIGGERS]
```

```
[TDSPACKETSIZE bytes]
[TRANSNAME trans_name]
[USEODBC | USEREPLICATIONUSER]
[XMLBUFSIZE bytes]
```

ALLOWUNUSEDCOLUMN | NOALLOWUNUSEDCOLUMN
Valid for Extract for Oracle. Controls whether Extract abends when it encounters a table with an unused column.
The default is ALLOWUNUSEDCOLUMN. When Extract encounters a table with an unused column, it continues processing and generates a warning.When using this parameter, either the same unused column must exist on the target or a source definitions file for the table must be specified to Replicat, so that the correct metadata mapping can be performed.
NOALLOWUNUSEDCOLUMN causes Extract to abend on unused columns.

ALLOWLOBDATATRUNCATE | NOALLOWLOBDATATRUNCATE
Valid for Replicat for DB2 LUW and MySQL. ALLOWLOBDATATRUNCATE prevents Replicat from abending when replicated LOB data is too large for a target CHAR, VARCHAR, BINARY or VARBINARY column and is applicable to target LOB columns only. or replicat of DB2 LUW, ALLOWLOBDATATRUNCATE prevents Replicat from abending when replicated LOB data is too large for a target LOB column. The LOB data is truncated to the maximum size of the target column *without any further error messages or warnings*.
NOALLOWLOBDATATRUNCATE is the default and causes Replicat to abend with an error message if the replicated LOB is too large.

BINDCHARFORBITASCHAR
Valid for DEFGEN, Extract, and Replicat for DB2 for i. Allows columns that are defined as CHAR or VARCHAR with CCSID 65535, or CHAR and VARCHAR FOR BIT DATA to be treated as if the field had a normal translatable encoding. The encoding is picked up from the job CCSID. When this option is in effect, DEFGEN does not indicate that the field is binary in the defs file.

CATALOGCONNECT | NOCATALOGCONNECT
Valid for Extract and Replicat for ODBC databases. By default, Oracle GoldenGate creates a new connection for catalog queries, but you can use NOCATALOGCONNECT to prevent that. On DB2 for z/OS, NOCATALOGCONNECT prevents Oracle GoldenGate from attempting multiple connections when the MVS DB2 initialization parameter mvsattachtype is set to CAF. Because CAF mode does not support multiple connections, it is possible that Oracle GoldenGate may issue commit locks on the system catalog tablespaces until it receives the commit for its open connection. To prevent commit locks, Oracle GoldenGate recommends using RRSAF (mvsattachtype=RRSAF), which supports multiple connections.

CONNECTIONPORT *port*
Valid for Replicat for multi-daemon MySQL. Specifies the TCP/IP port of the instance to which Replicat must connect. The minimum value is 1 and the default value is 3306.

DECRYPTPASSWORD *shared_secret algorithm* ENCRYPTKEY {*key_name* | DEFAULT}
Valid for Extract in classic capture mode (Oracle)
Specifies the shared secret (password) that decrypts the TDE key, which decrypts redo log data that was encrypted with Oracle Transparent Data Encryption (TDE). The TDE key is first encrypted in the Oracle server by using the shared secret as a key, and then it is delivered to Extract, which decrypts it by using the same shared secret.

The shared secret must be created in the Oracle Wallet or Hardware Security Module by the Oracle Server Security Officer. The only other person who should know the shared secret is the Oracle GoldenGate Administrator.
To use the decryption options, you must first generate the encrypted shared secret with the ENCRYPT PASSWORD command in GGSCI and create an ENCKEYS file.
Parameter options:

shared_secret
Is the encrypted shared secret (password) that is copied from the ENCRYPT PASSWORD command results.

algorithm
Specifies the encryption algorithm that was used to encrypt the password: AES128, AES192, AES256, or BLOWFISH. AES is not supported on DB2 on z/OS, and DB2 for i.

ENCRYPTKEY key_name
Specifies the logical name of a user-created encryption key in the ENCKEYS lookup file. Use if ENCRYPT PASSWORD was used with the KEYNAME *key_name* option. Requires an ENCKEYS file to be created on the local system.

ENCRYPTKEY DEFAULT
Directs Oracle GoldenGate to use a random key. Use if ENCRYPT PASSWORD was used with the KEYNAME DEFAULT option.

For more information about configuring Extract to support TDE, see Configuring Capture in Integrated Mode in *Using Oracle GoldenGate for Oracle Database*.
For more information about Oracle GoldenGate encryption options, including ENCKEYS, see Encrypting Data with the ENCKEYS Method in the *Administering Oracle GoldenGate*.

DEFERREFCONST
Valid for nonintegrated Replicat for Oracle. Sets constraints to DEFERRABLE to delay the checking and enforcement of cascade delete and cascade update referential integrity constraints by the Oracle target database until the Replicat transaction is committed. At that point, if there are constraint violations, an error is generated. Integrated Replicat does not require disabling of referential constraints on the target system.
You can use DEFERREFCONST instead of disabling the constraints on the target tables or setting them to DEFERRED. When used, DEFERREFCONST defers both DEFERABLE and NOT DEFERABLE constraints. DEFERREFCONST applies to every transaction that is processed by Replicat. DEFERREFCONST is valid for Oracle Database 12*c*, 11*g* (11.2.0.2), and later 11*g* R2 releases.
If used with an Oracle Database release that does not support this functionality, DEFERREFCONST is ignored without returning a notification to the Oracle GoldenGate log. To handle errors on the commit operation, you can use REPERROR at the root level of the parameter file and specify the TRANSDISCARD or TRANSEXCEPTION option.

> **Note:**
>
> Do not to use with DEFERREFCONST coordinated Replicat because there is no way to guarantee that related rows in parent and child tables are processed by same thread

DISABLECOMMITNOWAIT
Valid for Replicat for Oracle. Disables the use of asynchronous COMMIT by Replicat. An asynchronous COMMIT statement includes the NOWAIT option.
When DISABLECOMMITNOWAIT is used, Replicat issues a standard synchronous COMMIT (COMMIT with WAIT option).

DISABLELOBCACHING
Valid for nonintegrated Replicat for Oracle. Disables Oracle's LOB caching mechanism. By default, Replicat enables Oracle's LOB caching mechanism.

ENABLE_INSTANTIATION_FILTERING
Valid for Oracle. Enables automatic per table instantiation CSN filtering on tables imported using Oracle data pump or manually instantiated using the SET_INSTANTIATION_CSN command.

FETCHBATCHSIZE *records*
Valid for Extract for Oracle, DB2 for i, DB2 z/OS, SQL Server, and Teradata. Enables array fetches for initial loads to improve performance, rather than one row at a time. Valid values for Oracle, DB2 for i, DB2 z/OS, SQL Server, and Teradata are 0 through 1000000 records per fetch. Valid values for DB2 LUW are 1 through 1000000 records per fetch; zero (0) is *not* a valid value.
The default is 1000. Performance slows when batch size gets very small or very large. If the table contains LOB data, Extract reverts to single-row fetch mode, and then resumes batch fetch mode afterward.

FETCHCHECKFREQ *seconds*
Valid for Integrated Extract for Oracle. Specifies the number of seconds that Extract waits between each fetch check for the ADG to catch up. A low number improves latency though increases the number of queries of current_scn from v$database. The default is 3 seconds; the maximum is 120 seconds.

FETCHLOBS | NOFETCHLOBS
Valid for Extract for DB2 for z/OS and DB2 for LUW. Suppresses the fetching of LOBs directly from the database table when the LOB options for the table are set to NOT LOGGED. With NOT LOGGED, the value for the column is not available in the transaction logs and can only be obtained from the table itself. By default, Oracle GoldenGate captures changes to LOBs from the transaction logs. The default is FETCHLOBS.

FETCHRETRYCOUNT *number*
Valid for Integrated Extract for Oracle. Specifies the number of times that Extract tries before it reports ADG progress or the reason for no progress when waiting for the ADG to catch up. This value is multiplied with FETCHCHECKFREQ to determine approximately how often the ADG progress is reported.

FECHTIMEOUT *seconds* | NOFECHTIMEOUT
Valid for Integrated Extract for Oracle. Specifies the number of seconds that Extract after which it will abend when ADG makes no progress. No progress can be because

the MRP is not running or because it is not applying redo changes. When this occurs, the ADG database should be examined. The default is 30 seconds; valid values are 0 - 4294967295 (ub4 max value) seconds. NOFETCHTIMEOUT means never timeout (the same as FECHTIMEOUT 0) and seconds cannot be specified with it.

HOST *{DNS_name | IP_address}*
Valid for Replicat for multi-daemon MySQL. Specifies the DNS name or IP address of the system that hosts the instance to which Replicat must connect.

INTEGRATEDPARAMS(*parameter*[, ...])
Valid for Replicat for Oracle. Passes settings for parameters that control the database inbound server within the target Oracle database. Use this option only for an integrated Replicat. For more information about integrated Replicat and a list of supported inbound server parameters, see Choosing Capture and Apply Modes in *Using Oracle GoldenGate for Oracle Database*.

LIMITROWS | NOLIMITROWS
Valid for Replicat for MySQL, Oracle, and SQL Server. LIMITROWS prevents multiple rows from being updated or deleted by the same Replicat SQL statement when the target table does not have a primary or unique key.
LIMITROWS is the default. LIMITROWS and NOLIMITROWS apply globally to all MAP statements in a parameter file.
For MySQL, LIMITROWS uses a LIMIT 1 clause in the UPDATE or DELETE statement.
For Oracle targets, LIMITROWS (the default) must be used. It uses either WHERE ROWNUM = 1 or AND ROWNUM = 1 in the WHERE clause.
For SQL Server, LIMITROWS uses a SET ROWCOUNT 1 clause before the UPDATE or DELETE statement.
NOLIMITROWS permits multiple rows to be updated or deleted by the same Replicat SQL statement.

LOBBUFSIZE *bytes*
Valid for Extract for Oracle. Determines the memory buffer size in bytes to allocate for each embedded LOB attribute that is in an Oracle object type. Valid values are from 1024 and 104857600 bytes. The default is 1048576 bytes.
If the length of embedded LOB exceeds the specified LOBBUFSIZE size, an error message similar to the following is generated:

```
GGS ERROR    ZZ-0L3  Buffer overflow, needed: 2048, allocated: 1024.
```

LOBWRITESIZE *bytes*
Valid for nonintegrated Replicat for Oracle. Specifies a fragment size in bytes for each LOB that Replicat writes to the target database. The LOB data is stored in a buffer until this size is reached. Because LOBs must be written to the database in fragments, writing in larger blocks prevents excessive I/O. The higher the value, the fewer I/O calls that are made by Replicat to the database server to write the whole LOB to the database.
Specify a multiple of the Oracle LOB fragment size. A given value will be rounded up to a multiple of the Oracle LOB fragment size, if necessary. The default LOB write size is 32k if DBOPTIONS NOSKIPTEMPLOB is specified, or 1MB if DBOPTIONS SKIPTEMPLOB is specified. Valid values are from 2,048 bytes to 2,097,152 bytes (2MB).
By default, Replicat enables Oracle's LOB caching mechanism. To disable Oracle's LOB caching, use the DISABLELOBCACHING option of DBOPTIONS.

SESSIONPOOLMAX *max_value*
Valid for Extract in integrated mode for Oracle. Sets a maximum value for the number of sessions in the OCI Session Pool, which is used by Extract for fetching from a container database. The default value is 10 sessions. Must be specified before the USERID or USERIDALIAS parameter; otherwise will be ignored and the default will be used.

SESSIONPOOLMIN *min_value*
Valid for Extract in integrated mode for Oracle. Sets a minimum value for the number of sessions in the OCI Session Pool, which is used by Extract for fetching from a container database. The default value is 2 sessions. Must be specified before the USERID or USERIDALIAS parameter; otherwise will be ignored and the default will be used.

SESSIONPOOLINCR *increment_value*
Valid for Extract in integrated mode for Oracle. Sets a value for the number of incremental sessions that can be added to the OCI Session Pool, which is used by Extract for fetching from a container database. The default value is 2 sessions. Must be specified before the USERID or USERIDALIAS parameter; otherwise will be ignored and the default will be used.

SETTAG [*tag_value* | NULL
Valid for Replicat for Oracle. Sets the value for an Oracle redo tag that will be used to identify the transactions of the associated Replicat in the redo log. A redo tag also can be used to identify transactions other than those of Replicat.
Use this option to prevent cycling (loop-back) of Replicat the individual records in a bi-directional configuration or to filter other transactions from capture. The default SETTAG value is 00 and is limited to 2K bytes. A valid value is any single Oracle Streams tag. A tag value can be up to 2000 hexadecimal digits (0-9 A-F) long. Transactions in the redo that are marked with the specified tag can be filtered by an Extract that has the TRANLOGOPTIONS parameter with the EXCLUDETAG option set to the *tag_value*. Use tag-based filtering to prevent cycling (loop-back) of Replicat transactions in a bi-directional configuration or to filter other transactions from capture. For more information, see TRANLOGOPTIONS.
You can disable the tagging of DDL by using the DDLOPTIONS parameter with the NOTAG option.

hex_value
A hexadecimal value from 0 through F. The default value is 00. The following are valid examples:

```
DBOPTIONS SETTAG 00112233445566778899AABBCCDDEEFF
DBOPTIONS SETTAG 00112233445566778899aabbccddeeff
DBOPTIONS SETTAG 123
```

NULL
Disables tag-based filtering for the associated Replicat.

SKIPTEMPLOB | NOSKIPTEMPLOB
Valid for Replicat for Oracle Database versions 11*g* and 12*c*. Controls how LOBs are applied to a target Oracle database. The default of SKIPTEMPLOB .
SKIPTEMPLOB improves performance by directly writing LOB data to the target LOB column. Replicat creates a SQL statement with an empty LOB value and returns the LOB locator to the bind variable. After the SQL statement is executed successfully, the LOB data is written directly to the LOB column using the returned LOB locator.

NOSKIPTEMPLOB uses a temporary LOB in the SQL statement. Replicat declares a bind variable within SQL statement and associates a temporary LOB, then writes to the temporary LOB. The Oracle Database applies the LOB column data from the temporary LOB.
SKIPTEMPLOB applies to INSERT and UPDATE operations that contain LOB data. It does not apply if the table has a functional index with a LOB column, if the LOB data is NULL, empty, or stored inline. It does not apply to partial LOB operations.
SKIPTEMPLOB causes Replicat to generate/perform 1 DML+ n LOB_WRITE (piece-wise) operations when updating/inserting a row with LOB columns. However, SKIPTEMPLOB should not be used with FETCHPARTIALLOB (an Extract Parameter) because it results in excessive fetching.
NOSKIPTEMPLOB is provided for backward compatibility; otherwise the default of SKIPTEMPLOB should be retained.

SOURCE_DB_NAME *src_dbase_global_name*
Valid for Oracle. Indicates the Global Name of the Trail Source Database.
It is used to query the relevant instantiation information when DBOPTIONS ENABLE_INSTANTIATION_FILTERING is enabled. This option is optional for instantiation filtering in a 12.2. trail file with metadata enabled.
When the source has no DOMAIN, do not specify a DOMAIN for the downstream database.

SPTHREAD | NOSPTHREAD
Valid for Extract and Replicat for SQL Server. Creates a separate database connection thread for stored procedures. The default is NOSPTHREAD.

SQLMODE
With this option enabled, the sql_mode variable is set to to 'ANSI_QUOTES' (set sql_mode = 'ANSI_QUOTES'). Treat the double quotes (") as an identifier quote character (like the ` quote character) and not as a string quote character. You can still use ` to quote identifiers with this mode enabled. With ANSI_QUOTES enabled, you cannot use double quotes (") to quote literal strings, because it is interpreted as an identifier.
For more information, see Server SQL Modes.

SUPPRESSTEMPORALUPDATES
Valid for DB2 LUW 10.1 FixPack 2 and greater replication of temporal table.
Use SUPPRESSTEMPORALUPDATES to replicate system-period and bitemporal tables along with associated history tables. Oracle GoldenGate replicates the row begin, row end, and transaction start id columns along with the other columns of the table. You must ensure that the database instance has the execute permission to run the SYSPROC.SET_MAINT_MODE_RECORD_NO_TEMPORALHISTORY¿ stored procedure at the apply side.
By default, Oracle GoldenGate does not replicate row begin, row end, and transaction start id columns. To preserve the original values of these columns, implement one of the followings options.

- Add extra timestamp columns in the target temporal table and map the columns accordingly.
- Use a non-temporal table at the apply side and map the columns accordingly.

Replication in Heterogeneous Environment:
In heterogeneous environments where there is no temporal tables at the apply side, you need to set the row begin, row end and transaction start id columns value. These source columns will have timestamp values that the target database may not

support. You should first use the map conversion functions to convert these values into the format that target database supports, and then map the columns accordingly. For example, MySQL has a DATETIME range from `1000-01-01 00:00:00.000000' to '9999-12-31 23:59:59.999999¿'. You cannot replicate a DB2 LUW timestamp value of `0001-01-01-00.00.00.000000000000¿ to MySQL. To replicate such values you must convert this value into the MySQL DATETIME format. For example, if a system-period or bitemporal table has the following timestamp column:

```
SYS_START
------------------------------
0001-01-01-00.00.00.000000000000
```

Then to replicate this column into MySQL, you would use the function colmap() as follows:

```
map <source_schema>.<source_table>, target <target_schema>.<target_table>
colmap(sys_start= @IF( ( @NUMSTR( @STREXT(sys_start,1,4))) > 1000, sys_start,
'1000-01-01 00.00.00.000000'));
```

Initial Load of Temporal Table:
Oracle GoldenGate supports initial load of temporal table as usual.
Take into account the following considerations with temporal table:

- Replication between system-period and application-period temporal table is not supported.
- Replication from a non-temporal table to a temporal table is not supported.
- Replication of system-period, bi-temporal tables, and SUPPRESSTEMPORALUPDATES with the INSERTALLRECORDS parameter is not supported.
- If any unique index is created for application-period temporal table using BUSINESS_TIME WITHOUT OVERLAPS for the target table, then the same unique index must be created for the source table.
- Bidirectional replication between temporal tables is advised only with the default.
- CDR is supported only with SUPPRESSTEMPORALUPDATES. There is no CDR support in bidirectional replication.
- By default, there are inconsistencies in row begin, row end, and transaction start id columns of the temporal tables when the source and target databases operate with different time zones. These timestamp columns of system-period and bitemporal tables are automatically populated by the respective database managers and will have values as per the respective time zones of the databases.
- Using the default with GETUPDATEBEFORES is in the replicate parameter file, you cannot use the row begin, row end, and transaction start id columns in any delta calculations. For example, taking before and after image of such columns in any kind of calculations is not possible. These columns can be used in delta calculations using SUPPRESSTEMPORALUPDATES.

SUPPRESSTRIGGERS | NOSUPPRESSTRIGGERS
Valid for Integrated Replicat and Classic Replicat for Oracle. Controls whether or not triggers are fired during the Replicat session. Provides an alternative to manually disabling triggers. (Integrated Replicat does not require disabling of triggers on the target system.)
SUPPRESSTRIGGERS is the default and prevents triggers from firing on target objects that are configured for replication with Oracle GoldenGate. SUPPRESSTRIGGERS

is valid for Oracle Database 12*c*, 11*g* (11.2.0.2), and later 11*g* R2 releases. SUPPRESSTRIGGERS is not valid for 11*g* R1.
To allow a specific trigger to fire, you can use the following SQLEXEC statement in the Replicat parameter file, where *trigger_owner* is the owner of the trigger and *trigger_name* is the name of the trigger.

```
SQLEXEC 'DBMS_DDL.SET_TRIGGER_FIRING_PROPERTY('"S1"','"MY_TRIGGER"',FALSE);'
```

> **Note:**
>
> Once this SQLEXEC is executed with FALSE, the trigger will continue to fire until the command is run again with a setting of TRUE.

NOSUPPRESSTRIGGERS allows target triggers to fire. To use [NO]SUPPRESSTRIGGERS, the Replicat user must have the privileges granted through the dbms_goldengate_auth.grant_admin_privilege package. This procedure is part of the Oracle database installation. See the database documentation for more information.
The USERID or USERIDALIAS parameter must precede a DBOPTIONS statement that contains SUPPRESSTRIGGERS or NOSUPPRESSTRIGGERS.

TRANSNAME *trans_name*
Valid for Replicat for SQL Server. Allows an individual Replicat to use a specific transaction name that is specified in the parameter file. The *trans_name* is the name of the transaction that the Replicat uses for target DML transactions and overrides the default ggs_repl transaction name when used.

USEODBC
Valid for Replicat for SQL Server. Configures Replicat to use ODBC to perform DML operations. The default is to use OLE DB. USEODBC cannot be used in conjunction with USEREPLICATIONUSER and will cause the Replicat to abend.

> **Note:**
>
> Replicat always uses ODBC to connect to the database catalog to obtain metadata.

USEREPLICATIONUSER
Valid for Replicat for SQL Server. Configures the Replicat connection to honor the NOT FOR REPLICATIONproperty, which is set on objects within the database, such as Identity columns, Foreign Key columns, Check constraints, and Triggers.
When the replication user is used, the following concerns must be addressed for their effect on data integrity:

- IDENTITY seeds on the target are not updated. A partitioning scheme is needed to avoid primary key violations unless the target is read-only.
- Foreign key constraints are not enforced.

- `ON UPDATE CASCADE, ON DELETE CASCADE` and triggers are disabled. This is beneficial to Replicat, since it prevents duplicate operations, but may not be appropriate for the target applications and might require modification to the code of the constraint or trigger to ensure data integrity.
- `CHECK` constraints are not enforced, so data integrity cannot be certain on the target.

By default, `USEREPLICATIONUSER` is disabled. The use of `USEREPLICATIONUSER` is only advised if delivery performance must be increased. `USEREPLICATIONUSER` cannot be used in conjunction with `USEODBC` and will cause the Replicat to abend.

`XMLBUFSIZE` *`bytes`*
Valid for Extract for Oracle. Sets the size of the memory buffer that stores XML data that was extracted from the `sys.xmltype` attribute of a `SDO_GEORASTER` object type. The default is 1048576 bytes (1MB). If the data exceeds the default buffer size, Extract will abend. If this occurs, increase the buffer size and start Extract again. The valid range of values is 1024 to 104857600 bytes.

Examples

Example 1

```
DBOPTIONS HOST 127.0.0.1, CONNECTIONPORT 3307
```

Example 2

```
DBOPTIONS DECRYPTPASSWORD AACAAAAAAAAAAAIALCKDZIRHOJBHOJUH ENCRYPTKEY DEFAULT
```

Example 3

```
DBOPTIONS TDSPACKETSIZE 2048
```

Example 4

```
DBOPTIONS FETCHBATCHSIZE 2000
```

Example 5

```
DBOOPTION XMLBUFSIZE 2097152
```

2.38 DDL

Valid For

Extract and Replicat

Description

Use the `DDL` parameter to:

- enable DDL support
- filter DDL operations
- configure a processing action based on a DDL record

When used without options, the `DDL` parameter performs no filtering, and it causes all DDL operations to be propagated as follows:

- As an Extract parameter, it captures all supported DDL operations that are generated on all supported database objects and sends them to the trail.
- As a Replicat parameter, it replicates all DDL operations from the Oracle GoldenGate trail and applies them to the target. This is the same as the default behavior without this parameter.

When used with options, the DDL parameter acts as a filtering agent to include or exclude DDL operations based on:

- scope
- object type
- operation type
- object name
- strings in the DDL command syntax or comments, or both

Only one DDL parameter can be used in a parameter file, but you can combine multiple inclusion and exclusion options to filter the DDL to the required level.

- The filtering options of the DDL parameter are valid for a primary Extract that captures from the transaction source, but not for a data-pump Extract.
- When combined, multiple filter option specifications are linked logically as AND statements.
- All filter criteria specified with multiple options must be satisfied for a DDL statement to be replicated.
- When using complex filtering criteria in a DDL parameter statement, it is recommended that you test your configuration in a test environment before using it in production.
- See Example 1, Example for more information.

Do not use the DDL parameter for:

- an Extract data pump
- a VAM-sort Extract (Teradata source databases)

These process types do not permit the mapping or conversion of DDL and will propagate DDL records automatically in pass-through mode. DDL that is performed on a source table (for example ALTER TABLE TableA...) will be applied by Replicat with the same table name (ALTER TABLE TableA). It cannot be mapped as ALTER TABLE TableB.

For additional information about how to use Oracle GoldenGate DDL support, see Configuring DDL Support in *Using Oracle GoldenGate for Oracle Database* or *Using Oracle GoldenGate for Heterogeneous Databases*, as applicable.

Syntax

```
DDL [
{INCLUDE | EXCLUDE}
  [, MAPPED | UNMAPPED | OTHER | ALL]
  [, OPTYPE type]
  [, OBJTYPE 'type']
  [, SOURCECATALOG catalog | ALLCATALOGS]
  [, ALLOWEMPTYOBJECT]
  [, ALLOWEMPTYOWNER]
```

```
  [, OBJNAME name]
  [, INSTR 'string']
  [, INSTRWORDS 'word_list']
  [, INSTRCOMMENTS 'comment_string']
  [, INSTRCOMMENTSWORDS 'word_list']
  [, STAYMETADATA]
  [, EVENTACTIONS (action)
]
[...]
```

DDL Filtering Options

The following are the syntax options for filtering and operating upon the DDL that is replicated by Oracle GoldenGate. These options apply to the `INCLUDE` and `EXCLUDE` clauses of the DDL parameter and other parameters that support DDL replication.

`INCLUDE | EXCLUDE`
Use `INCLUDE` or `EXCLUDE` to identify the beginning of an inclusion or exclusion clause.

- An inclusion clause contains filtering criteria that identifies the DDL that this parameter will affect.
- An exclusion clause contains filtering criteria that excludes specific DDL from this parameter.

The inclusion or exclusion clause must consist of the `INCLUDE` or `EXCLUDE` keyword followed by any valid combination of the other filtering options of the `DDL` parameter. If you use `EXCLUDE`, you must create a corresponding `INCLUDE` clause. For example, the following is invalid:

```
DDL EXCLUDE OBJNAME "hr".*
```

However, you can use either of the following:

```
DDL INCLUDE ALL, EXCLUDE OBJNAME "hr"."*"
DDL INCLUDE OBJNAME fin.* EXCLUDE OBJNAME "fin.ss"
```

An `EXCLUDE` takes priority over any `INCLUDE`s that contain the same criteria. You can use multiple inclusion and exclusion clauses.
Do not include any Oracle GoldenGate installed DDL objects in a `DDL` parameter, in a `TABLE` parameter, or in a `MAP` parameter, nor in a `TABLEEXCLUDE` or `MAPEXCLUDE` parameter. Make certain that wildcard specifications in those parameters do not include Oracle GoldenGate-installed DDL objects. These objects must not be part of the Oracle GoldenGate configuration, but the Extract process must be aware of operations on them, and that is why you must *not* explicitly exclude them from the configuration with an `EXCLUDE`, `TABLEEXCLUDE`, or `MAPEXCLUDE` parameter statement.

`MAPPED | UNMAPPED | OTHER | ALL`
Use `MAPPED`, `UNMAPPED`, `OTHER`, and `ALL` to apply `INCLUDE` or `EXCLUDE` based on the DDL operation scope.

- `MAPPED` applies `INCLUDE` or `EXCLUDE` to DDL operations that are of `MAPPED` scope. `MAPPED` filtering is performed before filtering that is specified with other `DDL` parameter options.
- `UNMAPPED` applies `INCLUDE` or `EXCLUDE` to DDL operations that are of `UNMAPPED` scope.

- OTHER applies INCLUDE or EXCLUDE to DDL operations that are of OTHER scope.
- ALL applies INCLUDE or EXCLUDE to DDL operations of all scopes.

DDL EXCLUDE ALL is a special processing option that maintains up-to-date object metadata for Oracle GoldenGate, while blocking the replication of the DDL operations themselves. You can use DDL EXCLUDE ALL when using a method other than Oracle GoldenGate to apply DDL to the target, but you want Oracle GoldenGate to replicate data changes to the target objects. It provides the current metadata to Oracle GoldenGate as objects change, thus preventing the need to stop and start the Oracle GoldenGate processes. The following special conditions apply to DDL EXCLUDE ALL:

- DDL EXCLUDE ALL does not require the use of an INCLUDE clause.
- When using DDL EXCLUDE ALL, you can set the WILDCARDRESOLVE parameter to IMMEDIATE to allow immediate DML resolution if required.

OPTYPE *type*
Use OPTYPE to apply INCLUDE or EXCLUDE to a specific type of DDL operation, such as CREATE, ALTER, and RENAME. For *type*, use any DDL command that is valid for the database. For example, to include ALTER operations, the correct syntax is:

```
DDL INCLUDE OPTYPE ALTER
```

OBJTYPE '*type*'
Use OBJTYPE to apply INCLUDE or EXCLUDE to a specific type of database object. For *type*, use any object type that is valid for the database, such as TABLE, INDEX, and TRIGGER. For an Oracle materialized view and materialized views log, the correct types are snapshot and snapshot log, respectively. Enclose the name of the object type within single quotes. For example:

```
DDL INCLUDE OBJTYPE 'INDEX'
DDL INCLUDE OBJTYPE 'SNAPSHOT'
```

For Oracle object type USER, do not use the OBJNAME option, because OBJNAME expects owner.object or *container.owner.object* whereas USER only has a schema.

SOURCECATALOG *catalog* | ALLCATALOGS
Use these options to specify how unqualified object names in an OBJNAME clause are resolved to the correct container. Use these options when the source database is an Oracle container database.
SOURCECATALOG specifies a default container for all of the object names that are specified in the same INCLUDE or EXCLUDE clause. To take effect, SOURCECATALOG must be specified before the OBJNAME specification. See "SOURCECATALOG" for more information including using statements that contain two-part names, where three-part object names are required to fully identify an object.
ALLCATALOGS specifies that all of the containers of the database should be considered when resolving object names that are specified in the same INCLUDE or EXCLUDE clause. ALLCATALOGS can be placed before or after the OBJNAME specification.
The following is the order of precedence that is given when there are different catalog specifications in a parameter file:

1. ALLCATALOGS in an INCLUDE or EXCLUDE clause overrides all SOURCECATALOG specifications in the INCLUDE or EXCLUDE clause and at the root of the parameter file, and it overrides the container specification of a fully qualified object name in the OBJNAME clause.
2. An explicit catalog specification in the OBJNAME clause overrides all instances of SOURCECATALOG (but not ALLCATALOGS).
3. SOURCECATALOG in an INCLUDE or EXCLUDE clause overrides the global SOURCECATALOG parameter that is specified at the root of the TABLE or MAP statement.
4. The global SOURCECATALOG parameter takes effect for any unqualified object names in OBJNAME clauses if the INCLUDE or EXCLUDE clause does not specify SOURCECATALOG or ALLCATALOGS.
5. In the absence of any of the preceding parameters, all catalogs are considered.

ALLOWEMPTYOBJECT
Use ALLOWEMPTYOBJECT to allow an OBJNAME specification to process DDL that contains no object name. For example:

```
DDL INCLUDE OBJNAME sch.* ALLOWEMPTYOBJECT
```

ALLOWEMPTYOWNER
Use ALLOWEMPTYOWNER to allow an OBJNAME specification to process DDL that contains no owner name. For example:

```
DDL INCLUDE OBJNAME pdb.sch.* ALLOWEMPTYOWNER
```

OBJNAME *name*
Use OBJNAME to apply INCLUDE or EXCLUDE to the fully qualified name of an object. To specify two-part and three-part object names and wildcards correctly, see *Administering Oracle GoldenGate*.
Enclose case-sensitive object names within double quote marks.
Case-insensitive example:

```
DDL INCLUDE OBJNAME accounts.*
```

Case-sensitive example:

```
DDL INCLUDE OBJNAME accounts."cust"
```

Do not use OBJNAME for the Oracle USER object, because OBJNAME expects *owner.object* or *container.owner.object*, whereas USER only has a schema.
When using OBJNAME with MAPPED in a Replicat parameter file, the value for OBJNAME must refer to the name specified with the TARGET clause of the MAP statement. For example, given the following MAP statement, the correct value is OBJNAME fin2.*.

```
MAP fin.exp_*, TARGET fin2.*;
```

In the following example, a CREATE TABLE statement executes as follows on the source:

```
CREATE TABLE fin.exp_phone;
```

That same statement executes as follows on the target:

```
CREATE TABLE fin2.exp_phone;
```

If a target owner is not specified in the MAP statement, Replicat maps it to the database user that is specified with the USERID or USERIDALIAS parameter.
For DDL that creates derived objects, such as a trigger, the value for OBJNAME must be the name of the base object, not the name of the derived object.
For example, to include the following DDL statement, the correct value is hr.accounts, not hr.insert_trig.

```
CREATE TRIGGER hr.insert_trig ON hr.accounts;
```

For RENAME operations, the value for OBJNAME must be the new table name. For example, to include the following DDL statement, the correct value is hr.acct.

```
ALTER TABLE hr.accounts RENAME TO acct;
```

INSTR *'string'*
Use INSTR to apply INCLUDE or EXCLUDE to DDL statements that contain a specific character string within the command syntax itself, but not within comments. For example, the following excludes DDL that creates an index.

```
DDL INCLUDE ALL EXCLUDE INSTR 'CREATE INDEX'
```

Enclose the string within single quotes. The string search is not case sensitive. INSTR does not support single quotation marks (' ') that are within the string, nor does it support NULL values.

INSTRCOMMENTS *'comment_string'*
(Valid for Oracle) Use INSTRCOMMENTS to apply INCLUDE or EXCLUDE to DDL statements that contain a specific character string within a comment, but not within the DDL command itself. By using INSTRCOMMENTS, you can use comments as a filtering agent.
For example, the following excludes DDL statements that include the string 'source only' in the comments.

```
DDL INCLUDE ALL EXCLUDE INSTRCOMMENTS 'SOURCE ONLY'
```

In this example, DDL statements such as the following are not replicated.

```
CREATE USER john IDENTIFIED BY john /*source only*/;
```

Enclose the string within single quotes. The string search is not case sensitive. You can combine INSTR and INSTRCOMMENTS to filter on a string in the command syntax and in the comments of the same DDL statement.
INSTRCOMMENTS does not support single quotation marks (' ') that are within the string, nor does it support NULL values.

INSTRWORDS *'word_list'*
Use INSTRWORDS to apply INCLUDE or EXCLUDE to DDL statements that contain the specified words.
For *word_list*, supply the words in any order, within single quotes. To include spaces, put the space (and the word, if applicable) in double quotes. Double quotes also can be used to enclose sentences.
All specified words must be present in the DDL for INSTRWORDS to take effect.
Example:

```
DDL INCLUDE OPTYPE ALTER OBJTEYP 'TABLE' INSTRWORDS 'ALTER CONSTRAINT
" xyz"'
```

This example matches the following DDL statements:

```
ALTER TABLE ADD CONSTRAINT xyz CHECK

ALTER TABLE DROP CONSTRAINT xyz
```

`INSTRWORDS` does not support single quotation marks (' ') that are within the string, nor does it support `NULL` values.

`INSTRCOMMENTSWORDS 'word_list'`
(Valid for Oracle) Works the same way as `INSTRWORDS`, but only applies to comments within a DDL statement, not the DDL syntax itself. By using `INSTRCOMMENTS`, you can use comments as a filtering agent.
`INSTRCOMMENTSWORDS` does not support single quotation marks (' ') that are within the string, nor does it support `NULL` values.
You can combine `INSTRWORDS` and `INSTRCOMMENTSWORDS` to filter on a string in the command syntax and in the comments of the same DDL statement.

`STAYMETADATA`
(Valid for Oracle). Prevents metadata from being captured by Extract or applied by Replicat.
When Extract first encounters DML on a table, it retrieves the metadata for that table. When DDL is encountered on that table, the old metadata is invalidated. The next DML on that table is matched to the new metadata so that the target table structure always is up-to-date with that of the source.
However, if you know that a particular DDL operation will not affect the table's metadata, you can use `STAYMETADATA` so that the current metadata is not retrieved or replicated. This is a performance improvement that has benefit for such operations as imports and exports, where such DDL as truncates and the disabling of constraints are often performed. These operations do not affect table structure, as it relates to the integrity of subsequent data replication, so they can be ignored in such cases. For example `ALTER TABLE ADD FOREIGN KEY` does not affect table metadata.
An example of how this can be applied selectively is as follows:

```
DDL INCLUDE ALL INCLUDE STAYMETADATA OBJNAME xyz
```

This example states that all DDL is to be included for replication, but only DDL that operates on object `xyz` will be subject to `STAYMETADATA`.
`STAYMETADATA` also can be used the same way in an `EXCLUDE` clause.
`STAYMETADATA` must be used the same way on the source and target to ensure metadata integrity.
When `STAYMETADATA` is in use, a message is added to the report file. DDL reporting is controlled by the `DDLOPTIONS` parameter with the `REPORT` option.
This same functionality can be applied globally to all DDL that occurs on the source by using the `@ddl_staymetadata` scripts:

- `@ddl_staymetadata_on` globally turns off metadata versioning.
- `@ddl_staymetadata_off` globally enables metadata versioning again.

This option should be used with the assistance of Oracle GoldenGate technical support staff, because it might not always be apparent which DDL affects object

metadata. If improperly used, it can compromise the integrity of the replication environment.

`EVENTACTIONS (action)`
Causes the Extract or Replicat process take a defined action based on a DDL record in the transaction log or trail, which is known as the *event record*. The DDL event is triggered if the DDL record is eligible to be written to the trail by Extract or a data pump, or to be executed by Replicat, as determined by the other filtering options of the `DDL` parameter. You can use this system to customize processing based on database events.
For *`action`*, see `EVENTACTIONS` under the `MAP` and `TABLE` parameters.
Guidelines for using `EVENTACTIONS` on DDL records:

- `CHECKPOINTBEFORE`: Since each DDL record is autonomous, the DDL record is guaranteed to be the start of a transaction; therefore, the `CHECKPOINT BEFORE` event action is implied for a DDL record.
- `IGNORE`: This option is not valid for DDL records. Because DDL operations are autonomous, ignoring a record is equivalent to ignoring the entire transaction.

`EVENTACTIONS` does not support the following DDL objects because they are derived objects:

- indexes
- triggers
- synonyms
- `RENAME` on a table and `ALTER TABLE RENAME`

In a Teradata configuration where Extract is configured in maximum protection mode, use `EVENTACTIONS` only in the VAM-sort Extract group. It is not supported by the primary Extract in this configuration because concurrent changes are not sorted in transaction order at this point in the processing stream. For more information, see Understanding What's Supported for Teradata in *Using Oracle GoldenGate for Heterogeneous Databases*.

Examples

Example 1 Combining DDL Parameter Options
The following is an example of how to combine the options of the `DDL` parameter.

```
DDL  &
INCLUDE UNMAPPED &
    OPTYPE alter &
    OBJTYPE 'table' &
    OBJNAME users.tab* &
INCLUDE MAPPED OBJNAME * &
EXCLUDE MAPPED OBJNAME temporary.tab
```

The combined filter criteria in this statement specify the following:

- `INCLUDE` all `ALTER TABLE` statements for tables that are not mapped with a `TABLE` or `MAP` statement (`UNMAPPED` scope), but only if those tables are owned by `users` and their names start with `tab`,
- `INCLUDE` all DDL operation types for all tables that are mapped with a `TABLE` or `MAP` statement (`MAPPED` scope),

- EXCLUDE all DDL operation types for all tables that are MAPPED in scope, but only if those tables are owned by temporary and only if their names begin with tab.

Example 2 Including an Event Action
The following example specifies an event action of REPORT for all DDL records.

```
DDL INCLUDE ALL EVENTACTIONS (REPORT)
```

Example 3 Using an Event Action on a Subset of DDL
The following example shows how EVENTACTIONS can be used on a subset of the DDL. All DDL is to be replicated, but only the DDL that is executed on explicitly named objects qualifies to trigger the event actions of REPORT and LOG.

```
DDL INCLUDE ALL &
    INCLUDE OBJNAME sales.t* EVENTACTIONS (REPORT) &
    INCLUDE OBJNAME fin.my_tab EVENTACTIONS (LOG) &
```

Example 4
The following example demonstrates the different ways to specify catalog names for DDL that is issued on objects in a source Oracle container database.

- This includes pdb1.sch1.obj1 and pdb2.sch2.obj2 for DDL processing.

  ```
  SOURCECATALOG pdb1
  DDL INCLUDE OBJNAME sch1.obj1 INCLUDE SOURCECATALOG pdb2 OBJNAME sch2.obj2
  ```

- This includes all objects with the name sch.obj in any catalog for DDL processing.

  ```
  DDL INCLUDE ALLCATALOGS OBJNAME sch.obj
  ```

- This also includes all objects with the name sch.obj in any catalog for DDL processing, because ALLCATALOGS overrides any other catalog specification.

  ```
  DDL INCLUDE ALLCATALOGS OBJNAME pdb.sch.obj
  ```

Example 5
The following shows the combined use of ALLOWEMPTYOBJECT and ALLOWEMPTYOWNER.

```
DDL INCLUDE pdb.*.* ALLOWEMPTYOWNER ALLOWEMPTYOBJECT
```

2.39 DDLERROR

Valid For

Extract and Replicat

Description

Use the DDLERROR parameter to handle DDL errors on the source and target systems. Options are available for Extract and Replicat.

DDLERROR for Extract

Use the Extract option of the DDLERROR parameter to handle errors on objects found by Extract for which metadata cannot be found.

Default

Abend

Syntax

```
DDLERROR [RESTARTSKIP number_of_skips] [RETRYDELAY seconds] [SKIPTRIGGERERROR
number_of_errors]
```

RESTARTSKIP *number_of_skips*
Causes Extract to skip and ignore a specific number of DDL operations on startup, to prevent Extract from abending on an error. By default, a DDL error causes Extract to abend so that no operations are skipped. Valid values are 1 to 100000.
To write information about skipped operations to the Extract report file, use the DDLOPTIONS parameter with the REPORT option.

SKIPTRIGGERERROR *number_of_errors*
(Oracle) Causes Extract to skip and ignore a specific number of DDL errors that are caused by the DDL trigger on startup. Valid values are 1 through 100000.
SKIPTRIGGERERROR is checked before the RESTARTSKIP option. If Extract skips a DDL operation because of a trigger error, that operation is not counted toward the RESTARTSKIP specification.

DDLERROR for Replicat

Use the Replicat options of the DDLERROR parameter to handle errors that occur when DDL is applied to the target database. With DDLERROR options, you can handle most errors in a default manner, for example to stop processing, and also handle other errors in a specific manner. You can use multiple instances of DDLERROR in the same parameter file to handle all errors that are anticipated.

Default

Abend

Syntax

```
DDLERROR
{error | DEFAULT} {response}
{INCLUDE inclusion_clause | EXCLUDE exclusion_clause}
[IGNOREMISSINGOBJECTS | ABENDONMISSINGOBJECTS]
[RETRYDELAY seconds]
```

{error | DEFAULT} {response}

error
Specifies an explicit DDL error for this DDLERROR statement to handle.

DEFAULT
Specifies a default *response* to any DDL errors for which there is not an explicit DDLERROR statement.

response
The action taken by Replicat when a DDL error occurs. Can be one of the following:

ABEND
Roll back the operation and terminate processing abnormally. ABEND is the default.

DISCARD
Log the offending operation to the discard file but continue processing subsequent DDL.

IGNORE
Ignore the error.

{INCLUDE *inclusion_clause* | EXCLUDE *exclusion_clause*}
Identifies the beginning of an inclusion or exclusion clause that controls whether specific DDL is handled or not handled by the DDLERROR statement. See "DDL Filtering Options" for syntax and usage.

[IGNOREMISSINGOBJECTS | ABENDONMISSINGOBJECTS]
Controls whether or not Extract abends when DML is issued on objects that could not be found on the target. This condition typically occurs when DDL that is not in the replication configuration is issued directly on the target, or it can occur when there is a discrepancy between the source and target definitions.

IGNOREMISSINGOBJECTS
Causes Replicat to skip DML operations on missing tables.

ABENDONMISSINGOBJECTS
Causes Replicat to abend on DML operations on missing tables.

[RETRYDELAY *seconds*]
Specifies the delay in seconds between attempts to retry a failed operation. The default is 10 seconds.

Examples

Example 1 DDLERROR Basic Example
In the following example, the DDLERROR statement causes Replicat to ignore the specified error, but not before trying the operation again three times at ten-second intervals. Replicat applies the error handling to DDL operations executed on objects whose names satisfy the wildcard of `tab*` (any user, any operation) except those that satisfy `tab1*`.

```
DDLERROR 1234 IGNORE RETRYOP MAXRETRIES 3 RETRYDELAY 10 &
INCLUDE ALL OBJTYPE TABLE OBJNAME tab* EXCLUDE OBJNAME tab1*
```

To handle all errors except that error, the following DDLERROR statement can be added.

```
DDLERROR DEFAULT ABEND
```

In this case, Replicat abends on DDL errors.

Example 2 Using Multiple DDLERROR Statements
The order in which you list DDLERROR statements in the parameter file does not affect their validity unless multiple DDLERROR statements specify the same error, without any additional qualifiers. In that case, Replicat only uses the first one listed. For example, given the following statements, Replicat will abend on the error.

```
DDLERROR 1234 ABEND
DDLERROR 5678 IGNORE
```

With the proper qualifiers, however, the previous configuration becomes a more useful one. For example:

```
DDLERROR 1234 ABEND INCLUDE OBJNAME tab*
DDLERROR 5678 IGNORE
```

In this case, because there is an `INCLUDE` statement, Replicat will abend only if an object name in an errant DDL statement matches wildcard `tab*`. Replicat will ignore errant operations that include any other object name.

2.40 DDLOPTIONS

Valid For

Extract and Replicat

Description

Use the `DDLOPTIONS` parameter to configure aspects of DDL processing other than filtering and string substitution. You can use multiple `DDLOPTIONS` statements, but using one is recommended. If using multiple `DDLOPTIONS` statements, make each of them unique so that one does not override the other. Multiple `DDLOPTIONS` statements are executed in the order listed in the parameter file.

Default

See the argument descriptions

Syntax

```
DDLOPTIONS
[, ADDTRANDATA {ABEND | RETRYOP RETRYDELAY seconds MAXRETRIES retries}
[, DEFAULTUSERPASSWORD password [algorithm [ENCRYPTKEY DEFAULT | ENCRYPTKEY
key_name]
[, CAPTUREGLOBALTEMPTABLE ]
[, DEFAULTUSERPASSWORDALIAS alias [DOMAIN domain] ]
[, GETAPPLOPS | IGNOREAPPLOPS]
[, GETREPLICATES | IGNOREREPLICATES]
[, IGNOREMAPPING]
[, MAPDERIVED | NOMAPDERIVED]
[, MAPSCHEMAS]
[, MAPSESSIONSCHEMA source_schema TARGET target_schema]
[, NOTAG]
[, PASSWORD algorithm ENCRYPTKEY {key_name | DEFAULT}]
[, REMOVECOMMENTS {BEFORE | AFTER}]
[, REPLICATEPASSWORD | NOREPLICATEPASSWORD]
[, REPORT | NOREPORT]
[, UPDATEMETADATA]
[, USEPASSWORDVERIFIERLEVEL {10|11}]
[, _USEOWNERFORSESSION]
```

`ADDTRANDATA {ABEND | RETRYOP RETRYDELAY seconds MAXRETRIES retries}`
Valid for Extract (Teradata)
Not supported when Classic Extract is reading from an Active Data Guard standby database because supplemental logging must be enabled on the primary database, which is read/write. Not supported for multitenant container databases. Supplemental logging must be enabled when using `ADD SCHEMATRANDATA`. This option should *only* be used when schema-level supplemental logging is not an option in your environment. No longer valid for Oracle and a warning is issued if used.

Use ADDTRANDATA to:

- Enable Oracle supplemental logging automatically for new tables created with a CREATE TABLE statement.
- Update supplemental logging for tables affected by an ALTER TABLE statement to add or drop columns.
- Update supplemental logging for tables that are renamed.
- Update supplemental logging for tables where unique or primary keys are added or dropped.

By default, ADDTRANDATA is disabled. The default for ADDTRANDATA when used without additional options is:

```
DDLOPTIONS ADDTRANDATA RETRYOP RETRYDELAY 10 MAXRETRIES 10
```

To use ADDTRANDATA functionality, Oracle GoldenGate, the database, and the appropriate tables must be configured for DDL capture. For Oracle, the Oracle GoldenGate DDL objects must be installed and configured. For more information, see Preparing the Database for Oracle GoldenGate in *Using Oracle GoldenGate for Oracle Database*.
For new tables created with CREATE TABLE, ADDTRANDATA produces the same results as the default ADD TRANDATA command in GGSCI by issuing the Oracle ALTER TABLE command with the ADD SUPPLEMENTAL LOG GROUP option. Oracle GoldenGate executes this command when the CREATE TABLE or ALTER TABLE is captured on the source. If you have special requirements for the supplemental logging, use the ADD TRANDATA command, not DDLOPTIONS ADDTRANDATA. By default, the ALTER TABLE statement that adds the supplemental logging is not replicated to the target unless the GETREPLICATES option is in use.
For renamed tables, ADDTRANDATA deletes the supplemental log group for the old table and creates it for the new one. If you do not use ADDTRANDATA and tables will be renamed, do the following to create the log group before doing the rename:

1. Drop the supplemental log group using the database interface or the DELETE TRANDATA command in GGSCI.

   ```
   DELETE TRANDATA table_name
   ```

2. Rename the table.
3. Create the new supplemental log group using the database interface or the ADD TRANDATA command in GGSCI.

   ```
   ADD TRANDATA table_name
   ```

There might be a lag between the time when an original DDL operation occurs and when the ADD TRANDATA takes effect. During this time, do not allow DML operations (insert, update, delete) on the affected table if the data is to be replicated; otherwise, it will not be captured. To determine when DML can be resumed after ADDTRANDATA, do the following:

1. Edit the Extract parameter file in GGSCI.

> **WARNING:**
>
> Do not use the VIEW PARAMS or EDIT PARAMS command to view or edit a parameter file that was created in a character set other than that of the local operating system. View the file from outside GGSCI; otherwise, the contents may become corrupted.

2. Add the REPORT option to DDLOPTIONS, then save and close the file.

   ```
   DDLOPTIONS [, DDLOPTIONS_option] [,...] , REPORT
   ```

3. Stop and start Extract to activate the parameter changes.

   ```
   STOP EXTRACT group_name
   START EXTRACT group_name
   ```

4. View the Extract process report.

   ```
   VIEW REPORT group_name
   ```

5. Look for the ALTER TABLE statement that added the log group to the table, and make a note of the time that the command took effect. The entry looks similar to the following:

   ```
   Successfully added TRAN DATA for table with the key, table
   [MYSCHEMA1.MYTABLE], operation [ALTER TABLE "MYSCHEMA1"."MYTABLE"
   ADD SUPPLEMENTAL LOG GROUP "GGS_MYTABLE_53475" (MYID) ALWAYS  /*
   GOLDENGATE_DDL_REPLICATION */ ].
   ```

6. Permit DML operations on the new table.

The ADDTRANDATA options are:

ABEND
Causes Extract to abend.

RETRYOP
Causes Extract to try again based on RETRYDELAY and MAXRETRIES.

RETRYDELAY *seconds*
Sets the delay before Extract tries again. The default is 10 seconds. The maximum delay is 10,000 seconds.

MAXRETRIES *retries*
Sets the number of retries that Extract can make before abending. The default is 10 seconds. The maximum is 10,000 retries.

DEFAULTUSERPASSWORD *password* [*algorithm* ENCRYPTKEY {*key_name* | DEFAULT}]
Valid for Replicat. (Oracle only)
Can be used instead of the DEFAULTUSERPASSWORDALIAS option if an Oracle GoldenGate credential store is not being used. Specifies a different password for a replicated {CREATE | ALTER} USER *name* IDENTIFIED BY *password* statement from the one used in the source statement. Replicat will replace the placeholder that Extract writes to the trail with the specified password. When using DEFAULTUSERPASSWORD, use the NOREPLICATEPASSWORD option of DDLOPTIONS for Extract.
DEFAULTUSERPASSWORD *password* without options specifies a clear-text password. If the password is case-sensitive, type it that way.

> **Note:**
>
> Replication of CREATE | ALTER PROFILE will fail as the profile/password verification function must exist in the SYS schema. To replicate these DDLs successfully, password verification function must be created manually on both source/target(s) since DDL to SYS schema is excluded.

Use the following options if the password was encrypted with the `ENCRYPT PASSWORD` command in GGSCI:

algorithm
Specifies the encryption algorithm that was used to encrypt the password with the `ENCRYPT PASSWORD` command: `AES128`, `AES192`, `AES256`, or `BLOWFISH`. Use AES unless Blowfish is required for backward compatibility. AES is more secure than Blowfish.

ENCRYPTKEY *key_name*
Specifies the logical name of a user-created encryption key in the `ENCKEYS` lookup file. Use if `ENCRYPT PASSWORD` was used with the `KEYNAME` *`key_name`* option, and specify the same key name.

ENCRYPTKEY DEFAULT
Directs Oracle GoldenGate to use a random key. Use if `ENCRYPT PASSWORD` was used with the `KEYNAME DEFAULT` option.

CAPTUREGLOBALTEMPTABLE
Valid for Oracle
Allows Global Temporary Tables (GTT) DDLs to be visible to Extract so that they can be replicated. By default, GTT DDLs are not visible to Extract so using `CAPTUREGLOBALTEMPTABLE` you can set Extract to include GTT DDLs that then can be filtered by the DDL statement and if passed, written to the trail. The GTT DDLs are included in Replicat, if present in trail, and are filtered by the DDL statement then if they are passed they are executed.
For trigger-version of Extract, this option is set to false regardless of whether the table is GTT or not.

DEFAULTUSERPASSWORDALIAS *alias* [DOMAIN *domain*]
Valid for Replicat. (Oracle only)
Can be used instead of the `DEFAULTUSERPASSWORD` option if an Oracle GoldenGate credential store is being used. Specifies the alias of a credential whose password replaces the one in the `IDENTIFIED BY` clause of a replicated `CREATE USER` or `ALTER USER` statement. The alias is resolved to the encrypted password in the Oracle GoldenGate credential store. Replicat replaces the placeholder that Extract writes to the trail with the resolved password before applying the DDL to the target.
When using `DEFAULTUSERPASSWORDALIAS`, use the `NOREPLICATEPASSWORD` option of `DDLOPTIONS` for Extract.

alias
Specifies the alias of the credential whose password will be used for the replacement password. This credential must exist in the Oracle GoldenGate credential store. If you are not sure what alias to use, you can inspect the content of the credential store by issuing the `INFO CREDENTIALSTORE` command.

DOMAIN *domain*
Specifies the domain that is assigned to the specified user in the credential store.

GETAPPLOPS | IGNOREAPPLOPS
Valid for Extract. (Oracle only)
Controls whether or not DDL operations produced by business applications *except* Replicat are included in the content that Extract writes to a trail or file. GETAPPLOPS and IGNOREAPPLOPS can be used together with the GETREPLICATES and IGNOREREPLICATES options to control which DDL is propagated in a bidirectional or cascading configuration.

- For a bidirectional configuration, use GETAPPLOPS with IGNOREREPLICATES . You also must use the UPDATEMETADATA option.
- For a cascading configuration, use IGNOREAPPLOPS with GETREPLICATES on the systems that will be cascading the DDL operations to the target.

The default is GETAPPLOPS.

GETREPLICATES | IGNOREREPLICATES
Valid for Extract (Oracle only). Controls whether or not DDL operations produced by Replicat are included in the content that Extract writes to a trail or file. The default is IGNOREREPLICATES. For more information, see the GETAPPLOPS | IGNOREAPPLOPS options of DDLOPTIONS.

IGNOREMAPPING
Valid for Replicat. Disables the evaluation of name mapping that determines whether DDL is of MAPPED or UNMAPPED scope. This option improves performance in like-to-like DDL replication configurations, where source and target schema names and object names match, and where mapping functions are therefore unnecessary. With IGNOREMAPPING enabled, MAPPED or UNMAPPED scope cannot be determined, so all DDL statements are treated as OTHER scope. Do not use this parameter when source schemas and object names are mapped to different schema and object names on the target.

MAPDERIVED | NOMAPDERIVED
Valid for Replicat (Oracle and Teradata). Controls how derived object names are mapped.

MAPDERIVED
If a MAP statement exists for the derived object, the name is mapped to the name specified in that TARGET clause. Otherwise, the name is mapped to the name specified in the TARGET clause of the MAP statement that contains the base object. MAPDERIVED is the default.

NOMAPDERIVED
Prevents name mapping. NOMAPDERIVED overrides any explicit MAP statements that contain the name of the derived object.

For more information about how derived objects are handled during DDL replication, see the Managing the DDL Replication Environment in *Using Oracle GoldenGate for Oracle Database*.

MAPSCHEMAS
Valid for Replicat (Oracle and Teradata). Use only when MAPSESSIONSCHEMA is used.

- MAPSESSIONSCHEMA establishes a source-target mapping for session schemas and is used for objects whose schemas are not qualified in the DDL.
- MAPSCHEMAS maps objects that do have qualified schemas in the source DDL, but which do not qualify for mapping with MAP, to the same session-schema mapping as in MAPSESSIONSCHEMA. Examples of such objects are the Oracle CREATE TABLE AS SELECT statement, which contains a derived object in the AS SELECT clause, or the Teradata CREATE REPLICATION RULESET statement.

This mapping takes place after the mapping that is specified in the MAP statement.
As an example, suppose the following DDL statement is issued on a source Oracle database:

```
create table a.t as select from b.t;
```

Suppose the MAP statement on the target is as follows:

```
MAP a.*, TARGET c.*;
DDLOPTIONS MAPSESSIONSCHEMA b, TARGET b1, MAPSCHEMAS
```

As a result of this mapping, Replicat issues the following DDL statement on the target:

```
create table c.t as select from b1.t;
```

- The base table gets mapped according to the TARGET clause (to schema c).
- The qualified derived object (table t in SELECT FROM) gets mapped according to MAPSESSIONSCHEMA (to schema b1) because MAPSCHEMAS is present.

Without MAPSCHEMAS, the derived object would get mapped to schema c (as specified in the TARGET clause), because MAPSESSIONSCHEMA alone only maps unqualified objects.

MAPSESSIONSCHEMA *source_schema* TARGET *target_schema*
Valid for Replicat (Oracle only). Enables a source session schema to be mapped to (transformed to) a different session schema on the target.

- *source_schema* is the session schema that is set with ALTER SESSION set CURRENT_SCHEMA on the source.
- *target_schema* is the session schema that is set with ALTER SESSION set CURRENT_SCHEMA on the target.

Wildcards are not supported. You can use multiple MAPSESSIONSCHEMA parameters to map different schemas.
MAPSESSIONSCHEMA overrides any mapping of schema names that is based on master or derived object names
See the example at the end of this topic for usage.
See also MAPSCHEMAS.

NOTAG
Valid for Replicat
Prevents the tagging of DDL that is applied by Replicat with a redo tag (either the default tag '00' or one set with the DBOPTIONS parameter with the SETTAG option).
Use this option for bidirectional configurations where GETREPLICATES is used and DDL applied by Replicat must be captured back by Extract for a metadata refresh.

PASSWORD *algorithm* ENCRYPTKEY {*key_name* | DEFAULT}
Valid for Extract (Oracle only)

Directs Extract to encrypt all passwords in source DDL before writing the DDL to the trail.

algorithm
Specifies the encryption algorithm to be used to encrypt the password. Valid values are `AES128`, `AES192`, `AES256`, or `BLOWFISH`. Use AES unless Blowfish is required for backward compatibility. AES is more secure than Blowfish.

`ENCRYPTKEY` ***`key_name`***
Specifies the logical name of a user-created encryption key in an `ENCKEYS` lookup file.

`ENCRYPTKEY DEFAULT`
Directs Oracle GoldenGate to use a random key.

`REMOVECOMMENTS {BEFORE | AFTER}`
(Optional) Valid for Extract and Replicat (Oracle only). Controls whether or not comments are removed from the DDL operation. By default, comments are not removed, so that they can be used for string substitution with the `DDLSUBST` parameter. See "DDLSUBST" for more information.

`REMOVECOMMENTS BEFORE`
Removes comments before the DDL operation is processed by Extract or Replicat. They will not be available for string substitution.

`REMOVECOMMENTS AFTER`
Removes comments after they are used for string substitution. This is the default behavior if `REMOVECOMMENTS` is not specified.

`REPLICATEPASSWORD | NOREPLICATEPASSWORD`
Valid for Extract (Oracle only). Applies to the password in a `{CREATE | ALTER} USER` *`user`* `IDENTIFIED BY` *`password`* command.

- By default (`REPLICATEPASSWORD`), Oracle GoldenGate uses the source password in the target `CREATE` or `ALTER` statement.
- To prevent the source password from being sent to the target, use `NOREPLICATEPASSWORD`.

When using `NOREPLICATEPASSWORD`, specify a password for the target DDL statement by using a `DDLOPTIONS` statement with the `DEFAULTUSERPASSWORD` or `DEFAULTUSERPASSWORDALIAS` option in the Replicat parameter file.

`REPORT | NOREPORT`
Valid for Extract and Replicat (Oracle and Teradata). Controls whether or not expanded DDL processing information is written to the report file. The default of `NOREPORT` reports basic DDL statistics. `REPORT` adds the parameters being used and a step-by-step history of the operations that were processed.

`UPDATEMETADATA`
Valid for Replicat (Oracle only). Use in an active-active bi-directional configuration. This parameter notifies Replicat on the system where DDL originated that this DDL was propagated to the other system, and that Replicat should now update its object metadata cache to match the new metadata. This keeps Replicat's metadata cache synchronized with the current metadata of the local database.

USEPASSWORDVERIFIERLEVEL {10|11}
Only valid in an Oracle to Oracle configuration. Checks if the password verifier being sent in a DDL `CREATE USER` statement requires modifying. The reason for this check is because Oracle has different password verifiers, depending on the database version:

- 10g: A weak verifier kept in `user$.password`.
- 11g: The SHA-1 verifier.
- 12c: The SHA-2 and HTTP digest verifiers.

The SHA-1, SHA-2 and HTTP verifiers are captured in `user$.spare4` in the format of: `'S:<SHA-1-verifier>;H:<http-verifier>;T:<SHA-2-verifier>'`. Integrated Extract returns the following DDL in 12c for create user DDL statements:

- In 12.0.1.0 it returns: `CREATE USER` *`username`* `IDENTIFIED BY VALUES` 'S:*`SHA-1`*;H:*`http`*;*`weak`*'.
- In 12.0.2.0 and later it returns: `CREATE USER` *`username`* `IDENTIFIED BY VALUES` 'S:*`SHA-1`*;H:*`http`*;T:*`SHA-2;weak`*'.

If Replicat runs against Oracle 12*c*, these forms of `CREATE USER` are handled at the RDBMS level, but if Replicat runs against Oracle 10*g* or 11, these forms are not handled by the RDBMS. Oracle 10*g* only accepts the weak verifier, whereas Oracle 11*g* only accepts the `S:`*`SHA-1`* and weak verifiers.
To allow the `CREATE USER` DDL generated for an Extract connected to Oracle 12*c* to work with a Replicat connected to Oracle 10*g* or 11*g*, this parameter can be used to filter out the unwanted verifiers, as follows:

- If `USEPASSWORDVERIFIERLEVEL` is set to 10, everything except the weak verifier is filtered out of the `CREATE USER` DDL verification string.
- If `USEPASSWORDVERIFIERLEVEL` is set to 11, everything except the S:*`SHA-1`* and weak verifiers is filtered out of the `CREATE USER` DDL verification string.

Examples

Example 1

The following shows how `MAPSESSIONSCHEMA` works to allow mapping of a source session schema to another schema on the target.
Assume the following DDL capture and mapping configurations in Extract and Replicat:
Extract:

```
DDL INCLUDE OBJNAME SRC.* INCLUDE OBJNAME SRC1.*
TABLE SRC.*;
TABLE SRC1.*;
DDL INCLUDE OBJNAME SRC.* INCLUDE OBJNAME SRC1.*
TABLE SRC.*;
TABLE SRC1.*;
```

Replicat:

```
DDLOPTIONS MAPSESSIONSCHEMA SRC TARGET DST
DDLOPTIONS MAPSESSIONSCHEMA SRC1 TARGET DST1
MAP SRC.*, TARGET DST.*;
MAP SRC1.*, TARGET DST1.*;
DDL INCLUDE OBJNAME DST.* INCLUDE OBJNAME DST1.*
```

Assume that the following DDL statements are issued by the logged-in user on the source:

```
ALTER SESION SET CURRENT_SCHEMA=SRC;
CREATE TABLE tab (X NUMBER);
CREATE TABLE SRC1.tab (X NUMBER) AS SELECT * FROM tab;
```

Replicat will perform the DDL as follows (explanations precede each code segment):

```
-- Set session to DST, because SRC.* is mapped to DST.* in MAP statement.
ALTER SESION SET CURRENT_SCHEMA=DST;
-- Create the first TAB table in the DST schema, using the DST session schema.
CREATE TABLE DST.tab (X NUMBER);
-- Restore Replicat schema.
ALTER SESSION SET CURRENT_SCHEMA=REPUSER
-- Set session schema to DST, per MAPSESSIONSCHEMA, so that AS SELECT succeeds.
ALTER SESION SET CURRENT_SCHEMA=DST;
-- Create the DST1.TAB table AS SELECT * FROM the first table (DST.TAB).
CREATE TABLE DST1.tab (X NUMBER) AS SELECT * FROM tab;
-- Restore Replicat schema.
ALTER SESSION SET CURRENT_SCHEMA=REPUSER
```

Without `MAPSESSIONSCHEMA`, the `SELECT * FROM TAB` would attempt to select from a non-existent `SRC.TAB` table and fail. The default is to apply the source schema to unqualified objects in a target DDL statement. The DDL statement in that case would look as follows and would fail:

```
-- Set session to DST, because SRC.* is mapped to DST.* in MAP statement.
ALTER SESION SET CURRENT_SCHEMA=DST;
-- Create the first TAB table in the DST schema, using the DST session schema.
CREATE TABLE DST.tab (X NUMBER);
-- Restore Replicat schema.
ALTER SESSION SET CURRENT_SCHEMA=REPUSER
-- Set session schema to SRC, because TAB in the AS SELECT is unqualified-- and
SRC is the source session schema.
ALTER SESION SET CURRENT_SCHEMA=SRC;
-- Create DST1.TAB AS SELECT * from SRC.TAB (SRC=current session schema).
CREATE TABLE DST1.tab (X NUMBER) AS SELECT * FROM tab;
-- SRC.TAB does not exist.
-- Abend with an error unless the error is handled by a DDLERROR statement.
```

Example 2
The following shows how to use `DEFAULTUSERPASSWORDALIAS` to specify a different password for a replicated `{CREATE | ALTER} USER` *`name`* `IDENTIFIED BY` *`password`* statement from the one used in the source statement. In this example, the alias `ddlalias` is in the `target` domain in the credential store.

```
DDLOPTIONS DEFAULTUSERPASSWORDALIAS ddlalias DOMAIN target
```

2.41 DDLSUBST

Valid For

Extract and Replicat

Description

Use the `DDLSUBST` parameter to substitute strings in a DDL operation. For example, you could substitute one table name for another or substitute a string within

comments. The search is not case-sensitive. To represent a quotation mark in a string, use a double quote mark.

Guidelines for Using DDLSUBST

- Do not use DDLSUBST to convert column names and data types to something different on the target. Changing the structure of a target object in this manner will cause errors when data is replicated to it. Likewise, do not use DDLSUBST to change owner and table names in a target DDL statement. Always use a MAP statement to map a replicated DDL operation to a different target object.
- DDLSUBST always executes after the DDL parameter, regardless of their relative order in the parameter file. Because the filtering executes first, use filtering criteria that is compatible with the criteria that you are using for string substitution. For example, consider the following parameter statements:

  ```
  DDL INCLUDE OBJNAME fin.*
  DDLSUBST 'cust' WITH 'customers' INCLUDE OBJNAME sales.*
  ```

  In this example, no substitution occurs because the objects in the INCLUDE and DDLSUBST statements are different. The fin-owned objects are included in the Oracle GoldenGate DDL configuration, but the sales-owned objects are not.
- You can use multiple DDLSUBST parameters. They execute in the order listed in the parameter file.
- For Oracle DDL that includes comments, do not use the DDLOPTIONS parameter with the REMOVECOMMENTS BEFORE option if you will be doing string substitution on those comments. REMOVECOMMENTS BEFORE removes comments before string substitution occurs. To remove comments, but allow string substitution, use the REMOVECOMMENTS AFTER option.
- There is no maximum string size for substitutions, other than the limit that is imposed by the database. If the string size exceeds the database limit, the Extract or Replicat process that is executing the operation abends.

Default

No substitution

Syntax

```
DDLSUBST 'search_string' WITH 'replace_string'
[INCLUDE inclusion_clause | EXCLUDE exclusion_clause]
```

'search_string'
The string in the source DDL statement that you want to replace. Enclose the string within single quote marks. To represent a quotation mark in a string, use a double quotation mark.

WITH
Required keyword.

'replace_string'
The string that you want to use as the replacement in the target DDL. Enclose the string within single quote marks. To represent a quotation mark in a string, use a double quotation mark.

INCLUDE *inclusion_clause* | EXCLUDE *exclusion_clause*
Specifies one or more INCLUDE and EXCLUDE statements to filter the DDL operations for which the string substitution rules are applied. See "DDL Filtering Options" for syntax and usage.

Examples

Example 1
The following replaces the string cust with the string customers for tables in the fin schema.

```
DDLSUBST 'cust' WITH 'customers'
INCLUDE ALL OBJTYPE 'table' OBJNAME fin.*
```

Example 2
The following substitutes a new directory only if the DDL command includes the word logfile. If the search string is found multiple times, the replacement string is inserted multiple times.

```
DDLSUBST '/file1/location1' WITH '/file2/location2' INCLUDE INSTR 'logfile'
```

Example 3
The following uses multiple DDLSUBST statements, which execute in the order shown.

```
DDLSUBST 'a' WITH 'b' INCLUDE ALL
DDLSUBST 'b' WITH 'c' INCLUDE ALL
```

The net effect of the preceding substitutes all a and b strings with c.

2.42 DDLRULEHINT

Valid For

GLOBALS

Description

Use the DDLRULEHINT parameter to add a RULE hint to the DDL trigger. For example you can add the RULE (/*+NO_UNNEST*/) hint to improve the performance of the trigger when performing SQL queries.

You can also use the define _ddl_rule_hint parameter in the params.sql file to add a hint. For example: define _ddl_rule_hint = '/*+NO_UNNEST*/'

Default

None

Syntax

```
DDLRULEHINT hint_syntax
```

hint_syntax
The syntax of the hint. Spaces are not permitted within the hint syntax.

Example

```
DDLRULEHINT /*+NO_UNNEST*/
```

2.43 DDLTABLE

Valid For

GLOBALS

Description

Use the `DDLTABLE` parameter to specify the name of the DDL history table, if other than the default of `GGS_DDL_HIST`. The DDL history table stores a history of DDL operations processed by Oracle GoldenGate.

The name of the history table must also be specified with the `ddl_hist_table` parameter in the `params.sql` script. This script resides in the root Oracle GoldenGate installation directory.

This parameter is only valid for an Oracle database in which the capture configuration uses the Oracle GoldenGate DDL trigger to support DDL replication. For more information about the Oracle GoldenGate DDL objects, see Installing Trigger-Based DDL Capture in *Using Oracle GoldenGate for Oracle Database*.

Default

`GGS_DDL_HIST`

Syntax

`DDLTABLE` *`table_name`*

`table_name`
The fully qualified name of the DDL history table. This can be a two-part name (*`schema.table`*) or a three-part name, if stored in a container database (*`container.schema.table`*).

Example

```
DDLTABLE GG_DDL_HISTORY
```

2.44 DECRYPTTRAIL

Valid For

Extract data pump and Replicat

Description

Use the `DECRYPTTRAIL` parameter to decrypt data in a trail or extract file. This parameter is required in the following cases:

- If the trail was encrypted with the master key and wallet method, use `DECRYPTTRAIL` for a data pump only if you want the data to be decrypted when written to the output trail of the data pump. Otherwise, this parameter is not needed. If the data-pump requires further processing of records, it decrypts automatically and then re-encrypts the data before writing it to the output trail.

Replicat always decrypts data automatically when the master key and wallet method is used.

- When DECRYPTTRAIL is used for a data pump, use the ENCRYPTTRAIL parameter before specifying any output trails that must be encrypted.
- If the trail was encrypted with the ENCKEYS method, use DECRYPTTRAIL for Replicat to decrypt the data before applying it to the target.

Data encryption is controlled by the ENCRYPTTRAIL | NOENCRYPTTRAIL parameters.

For Oracle, if you are using wallet based encryption DECRYPTTRAIL does not require a cipher because it is recorded in the trail file header.

Default

None

Syntax

```
DECRYPTTRAIL [{AES128 | AES192 | AES256}]
```

DECRYPTTRAIL
Valid without any other options only if the trail or file was encrypted with ENCRYPTTRAIL without options to use 256-key byte substitution.

{AES128 | AES192 | AES256}
Valid for master key and wallet method and ENCKEYS method. Specify the same AES cipher that was used in ENCRYPTTRAIL to encrypt the trail or file.

Example

Example 1
The following is an example of the ENCKEYS method.

```
DECRYPTTRAIL AES192
```

Example 2
The following is all that is needed to decrypt using the master key and wallet method.

```
DECRYPTTRAIL
```

2.45 DEFERAPPLYINTERVAL

Valid For

Replicat

Description

Use the DEFERAPPLYINTERVAL parameter to set an amount of time that Replicat waits before applying captured transactions to the target database. To determine when to apply the transaction, Replicat adds the delay value to the commit timestamp of the source transaction, as recorded in the local GMT time of the source system.

You can use DEFERAPPLYINTERVAL for such purposes as to prevent the propagation of erroneous changes made to the source data, to control data arrival across different time zones, and to allow time for other planned events to occur before the data is applied to the target. Note that by using DEFERAPPLYINTERVAL, you are purposely

building latency into the target data, and it should be used with caution if the target applications are time-sensitive.

To find out if Replicat is deferring operations, use the `SEND REPLICAT` command with the `STATUS` option and look for a status of `Waiting on deferred apply`.

> **Note:**
>
> If the `TCPSOURCETIMER` parameter is in use, it is possible that the timestamps of the source and target transactions could vary by a few seconds, causing Replicat to hold its transaction (and hence row locks) open for a few seconds. This small variance should not have a noticeable affect on performance.

Default

0 (no delay)

Syntax

```
DEFERAPPLYINTERVAL n unit
```

n
A numeric value for the amount of time to delay. The minimum delay time is the value that is set for the `EOFDELAY` parameter. The maximum delay time is seven days.

unit
The unit of time for the delay. Can be:

```
S | SEC | SECS | SECOND | SECONDS | MIN | MINS | MINUTE | MINUTES | HOUR |
HOURS | DAY | DAYS
```

Example

This example directs Replicat to wait ten hours before posting its transactions.

```
DEFERAPPLYINTERVAL 10 HOURS
```

If a transaction completes at 08:00:00 source GMT time, and the delay time is 10 hours, the transaction will be applied to the target at 18:00:00 target GMT time the same day.

2.46 DEFSFILE

Valid For

DEFGEN

Description

Use the `DEFSFILE` parameter to identify the name of the file to which DEFGEN will write data definitions. By default, the data definitions file is written in the character set of the local operating system. You can change the character set with the `CHARSET` option.

For more information about definitions files, see *Administering Oracle GoldenGate*.

Default

None

Syntax

```
DEFSFILE file_name [APPEND | PURGE] [CHARSET character_set] [FORMAT RELEASE
major.minor]
```

file_name
The relative or fully qualified file name. The file is created when you run DEFGEN.

APPEND
Directs DEFGEN to write new content (from the current run) at the end of any existing content, if the specified file already exists. If the definitions file already exists, but is of an older Oracle GoldenGate release version, you can set the `FORMAT RELEASE` option to the same version as the existing file to prevent errors. Otherwise, DEFGEN will try to add newer metadata features and abend. The following are the restrictions when using `APPEND`:

- If the existing data definitions file is in a format older than Oracle GoldenGate 11.2.1, DEFGEN appends the table definitions in the old format, where table and column names with multi-byte and special characters are not supported.
- If the existing data definitions file is in the newer format introduced in version 11.2.1, DEFGEN appends the table definitions in the existing character set of the file.
- If the existing file is from version 11.2 or earlier, it was written when DEFGEN did not support three-part object names and will cause an error if the new metadata contains three-part names. You can specify objects from an Oracle container database if you remove the container or catalog portion by using the `NOCATALOG` parameter in the DEFGEN parameter file.

PURGE
Directs DEFGEN to purge the specified file before writing new content from the current run. When using `PURGE`, you can overwrite an existing definitions file that was created by an older version of DEFGEN with newer metadata that supports newer features, such as three-part object names.

CHARSET *character_set*
Generates the definitions file in the specified character set. Without `CHARSET`, the default character set of the operating system is used. If `APPEND` mode is specified for a definitions file that is version 11.2.1 or later, `CHARSET` is ignored, and the character set of the existing definitions file is used.

FORMAT RELEASE *major.minor*
Specifies the metadata format of the definitions that are sent by DEFGEN to the definitions file. The metadata tells the reader process whether the file records are of a version that it supports. The metadata format depends on the version of the Oracle GoldenGate process. Older Oracle GoldenGate versions contain different metadata than newer ones. Use `FORMAT` when the definitions file will be used by a process that is of an older Oracle GoldenGate version than the current one.

- FORMAT is a required keyword.
- RELEASE specifies an Oracle GoldenGate release version. *major* is the major version number, and *minor* is the minor version number. The X.x must reflect a current or earlier, generally available (GA) release of Oracle GoldenGate. Valid values are 9.0 through the current Oracle GoldenGate X.x version number, for example 11.2 or 12.1. (If you use an Oracle GoldenGate version that is earlier than 9.0, specify either 9.0 or 9.5.)

  The release version is programmatically mapped back to an appropriate internal compatibility level. The default is the current version of the process that writes to this trail. Note that RELEASE versions earlier than 12.1 do not support three-part object names.

Example

```
DEFSFILE ./dirdef/orcldef CHARSET ISO-8859-11 FORMAT RELEASE 11.2
```

2.47 DISCARDFILE | NODISCARDFILE

Valid For

Extract and Replicat

Description

Use the DISCARDFILE parameter to do the following:

- Customize the name, location, size, and write mode of the discard file. By default, a discard file is generated whenever a process is started with the START command through GGSCI. To retain the default properties, a DISCARDFILE parameter is not required.
- Specify the use of a discard file for processing methods where the process starts from the command line of the operating system and a discard file is not created by default.

Use the NODISCARDFILE parameter to disable the use of a discard file. If NODISCARDFILE is used with DISCARDFILE, the process abends.

When using DISCARDFILE, use either the PURGE or APPEND option. Otherwise, you must specify a different discard file name before starting each process run, because Oracle GoldenGate will not write to an existing discard file without one of these instructions and will terminate.

See "DISCARDROLLOVER" for how to control how often the discard file is rolled over to a new file.

For more information about the discard file, see *Administering Oracle GoldenGate*.

Default

If a process is started with the START command in GGSCI, it generates a discard file as follows:

- The file is named after the process that creates it, with a .dsc extension. If the process is a coordinated Replicat, it generates one file per thread. Each file name is appended with the thread ID of the corresponding thread.

- The file is created in the `dirrpt` sub-directory of the Oracle GoldenGate installation directory.
- The maximum file size is 50 MB.
- At startup, if a discard file exists, it is purged before new data is written.
- The maximum filename is 250 characters including the directory.

When you start a process from the command line of the operating system, you should not generate a discard file by default.

Syntax

```
DISCARDFILE { [file_name]
[, APPEND | PURGE]
[, MAXBYTES n | MEGABYTES n] } |
NODISCARDFILE
```

DISCARDFILE
Indicates that the name or other attribute of the discard file is being changed.

file_name
The relative or fully qualified name of the discard file, including the actual file name. For a coordinated Replicat, specify a file name of up to five characters, because each file name is appended with the thread ID of the thread that writes it. To store the file in the Oracle GoldenGate directory, a relative path name is sufficient, because Oracle GoldenGate qualifies the name with the Oracle GoldenGate installation directory.

APPEND
Adds new content to existing content if the file already exists. If neither `APPEND` nor `PURGE` is used, you must specify a different discard file name before starting each process run.

PURGE
Purges the file before writing new content. If neither `PURGE` nor `APPEND` is used, you must specify a different discard file name before starting each process run.

MAXBYTES *n*
Sets the maximum size of the file in bytes. The valid range is from 1 to 4096967295. The default is 3000000. If the specified size is exceeded, the process abends.

MEGABYTES *n*
Sets the maximum size of the file in megabytes. The valid range is from 1 to 4096. The default is 3. If the specified size is exceeded, the process abends.

NODISCARDFILE
Prevents the process from creating a discard file.

Example

Example 1
This example specifies a non-default file name and extension, non-default write mode, and non-default maximum file size. This example shows how you could change the default properties of a discard file for an online (started through GGSCI) process or specify the use of a discard file for a process that starts from the command line of the operating system and has no discard file by default.

```
DISCARDFILE .dirrpt/discard.txt, APPEND, MEGABYTES 20
```

Example 2
This example changes only the write mode of the default discard file for an online process (started through GGSCI).

```
DISCARDFILE .dirrpt/finance.dsc, APPEND
```

Example 3
This example disables the use of a discard file for an online process (started through GGSCI).

```
NODISCARDFILE
```

2.48 DISCARDROLLOVER

Valid For

Extract and Replicat

Description

Use the DISCARDROLLOVER parameter to set a schedule for aging discard files. For long or continuous runs, setting an aging schedule prevents the discard file from filling up and causing the process to abend, and it provides a predictable set of archives that can be included in your archiving routine.

When the DISCARDROLLOVER age point is reached, a new discard file is created, and old files are renamed in the format of *GROUPn.extension*, where:

- *GROUP* is the name of the Extract or Replicat group.
- *n* is a number that gets incremented by one each time a new file is created, for example: myext0.dsc, myext1.dsc, myext2.dsc, and so forth.
- *extension* is the file extension, such as .dsc.

You can specify a time of day, a day of the week, or both. Specifying just a time of day (AT option) without a day of the week (ON option) generates a discard file at the specified time every day.

Discard files roll over at the start of a process run. However, if APPEND or PURGE is specified in DISCARDFILE parameter, then discard files don't roll over at the start of a process.

If the NODISCARDFILE parameter is used with the DISCARDROLLOVER parameter, the process abends.

For more information about the discard file, see *Administering Oracle GoldenGate*.

Default

Disabled. By default, discard files are rolled over when a process starts.

Syntax

```
DISCARDROLLOVER
{AT hh:mi |
ON day |
AT hh:mm ON day}
```

AT *hh:mi*
The time of day to age the file.
Valid values:

- *hh* is an hour of the day from 00 through 23.
- *mm* is minutes from 00 through 59.

ON *day*
The day of the week to age the file.
Valid values:

```
SUNDAY
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
```

They are not case-sensitive.

Examples

Example 1

```
DISCARDROLLOVER AT 05:30
```

Example 2

```
DISCARDROLLOVER ON friday
```

Example 3

```
DISCARDROLLOVER AT 05:30 ON friday
```

2.49 DYNAMICRESOLUTION | NODYNAMICRESOLUTION

Valid For

Extract and Replicat

Description

Use the DYNAMICRESOLUTION and NODYNAMICRESOLUTION parameters to control how table names are resolved.

DYNAMICRESOLUTION, the default, enables fast process startup when there are numerous tables specified in TABLE or MAP statements. To get metadata for transaction records that it needs to process, Oracle GoldenGate queries the database and then builds a record of the tables that are involved. DYNAMICRESOLUTION causes the record to be built one table at a time, instead of all at once. The metadata of any given table is added when Extract first encounters the object ID in the transaction log, while record-building for other tables is deferred until their object IDs are encountered. DYNAMICRESOLUTION is the same as WILDCARDRESOLVE DYNAMIC.

NODYNAMICRESOLUTION causes the entire object record (for all tables) to be built at startup, which can be time-consuming if the database is large. This option is not supported for Teradata. NODYNAMICRESOLUTION is the same as WILDCARDRESOLVE IMMEDIATE.

See "WILDCARDRESOLVE" for more information.

Default

```
DYNAMICRESOLUTION
```

Syntax

```
DYNAMICRESOLUTION | NODYNAMICRESOLUTION
```

2.50 EBCDICTOASCII

Valid For

Extract data pump and Replicat

Description

Use the `EBCDICTOASCII` parameter to convert character data in the input trail from EBCDIC to ASCII format when sending it to a DB2 target database on a z/OS system. This parameter can be specified to request conversion of all EBCDIC columns and user token data to ASCII. This parameter must precede the `SOURCEDB` parameter. This parameter is only needed if the input trail file was created by an Extract version prior to v10.0. It is ignored for all other cases, because the conversion is done automatically.

As of version 11.2.1, conversion is not allowed by a data pump.

Default

None

Syntax

```
EBCDICTOASCII
```

2.51 ENABLEMONITORING

Valid For

GLOBALS

Description

Use the `ENABLEMONITORING` parameter to enable the monitoring of Oracle GoldenGate instances from Oracle GoldenGate Monitor and to collect trend data for Performance Metrics Server. It directs Manager to publish the monitoring points that provide status and other information to the Oracle GoldenGate Monitor clients.

Performance Metrics Server is used to monitor processes or services and collect statistics. To bring up the `PMSRVR GLOBALS` parameter, the `ENABLEMONITORING UDP` parameter must be set before starting the server in legacy. For Oracle GoldenGate MA it is done by default.
Before you enable monitoring on any given platform, see Introduction to Oracle GoldenGate Monitoring in the *Administering Oracle GoldenGate Monitor* for your database to make certain that the operating system is supported.

> **Note:**
>
> When monitoring is enabled on a UNIX system for a high number of Oracle GoldenGate processes (approximately 400), the system-imposed limit on the maximum amount of allowed shared memory may be exceeded. The message returned by Manager is similar to this:
>
> ```
> WARNING OGG-01934  Datastore repair failed" reported during "start...
> ```
>
> If this occurs, increase the kernel parameter `kernel.shmall` by eight times the default for the operating system.

Default

Disabled

Syntax

```
ENABLEMONITORING [ UDP ]
ENABLEMONITORING [ UDP ] [ UDPPORT portnumber ] [ HTTPPORT portnumber ]
```

`UDPPORT` *`portnumber`*
Valid with `UDP` for monitoring with a Performance Metrics Server (`PMSRVR`). Not valid for the BDB or LMDB monitoring modes.
The UDP listening port. It is optional. If provided, it overrides the existing `GLOBALS` parameter,`REPOUDPPORT`. If not provided, it uses the value of `REPOUDPPORT` as the port number.
You can change the UDP port of a `PMSRVR` in a secure deployment by adding the `repoUPDPORT` parameter to the `GLOBALS` file. For more information on configuring the UDP and TCP ports for PMSRVR, see How to Create Secure or Non-Secure Deployments.

`HTTPPORT` *`portnumber`*
Valid with `UDP` for monitoring with a Performance Metrics Server. Not valid for the BDB or LMDB monitoring modes.
The HTTP listening port for the service. It is optional. If not provided, 9004 is the default port number.

2.52 ENABLE_HEARTBEAT_TABLE | DISABLE_HEARTBEAT_TABLE

Valid For

Extract, Replicat, and GLOBALS

Description

The `ENABLE_HEARTBEAT_TABLE` and `DISABLE_HEARTBEAT_TABLE` commands specify whether the Oracle GoldenGate process will be handling records from `GG_HEARTBEAT` table or not. When specified as a `GLOBALS`, it is true for the entire installation unless overridden by a specific process.

Default

ENABLE_HEARTBEAT_TABLE

Syntax

ENABLE_HEARTBEAT_TABLE | DISABLE_HEARTBEAT_TABLE

ENABLE_HEARTBEAT_TABLE
Enables Oracle GoldenGate processes to handle records from a GG_HEARTBEAT table. This is the default.

DISABLE_HEARTBEAT_TABLE
Disables Oracle GoldenGate processes from handing records from a GG_HEARTBEAT table

2.53 ENCRYPTTRAIL | NOENCRYPTTRAIL

Valid For

Extract

Description

Use the ENCRYPTTRAIL and NOENCRYPTTRAIL parameters to control whether Oracle GoldenGate encrypts or does not encrypt data that is written to a trail or extract file.

ENCRYPTTRAIL supports the following encryption methods:

- **Master key and wallet method**: Generate a one-time AES key for each trail file and uses it to encrypt the contents. Then, the one-time key is encrypted by the master-key and stored in the trail file header.
- **ENCKEYS method**: Generate a AES encryption key, store it under a given name in an ENCKEYS file, and configure Oracle GoldenGate to use that key to directly encrypt or decrypt the contents of the trail file.

You can use encryption for local and remote trails by specifying the EXTTRAIL and RMTTRAIL parameters in your Extract parameter file:

ENCRYPTTRAIL and NOENCRYPTTRAIL are trail or file-specific. One affects all subsequent trail or extract file specifications in the parameter file until the other parameter is encountered. The parameter must be placed before the parameter entry for the trail that it will affect.

ENCRYPTTRAIL and NOENCRYPTTRAIL cannot be used when FORMATASCII is used to write data to a file in ASCII format. The trail or file must be written in the default Oracle GoldenGate canonical format.

ENCRYPTTRAIL encrypts the trail data across all data links and within the files themselves. Only the data blocks are encrypted. User tokens are not encrypted.

Default

NOENCRYPTTRAIL

Syntax

```
ENCRYPTTRAIL [{AES128 | AES192 | AES256}] | NOENCRYPTTRAIL]
```

ENCRYPTTRAIL
ENCRYPTTRAIL without options specifies 256-key byte substitution AES256 as the default for all database types except the iSeries, z/OS, and NonStop platforms because Advanced Encryption Standard (AES) encryption is not supported on those platforms.

{AES128 | AES192 | AES256}
Specifies the AES encryption key length to use. This is a symmetric-key encryption standard that is used by governments and other organizations that require a high degree of data security. This option is not supported by the iSeries, z/OS, and NonStop platforms.
For both the master key and wallet method and the ENCKEYS method, you must specify one of the AES ciphers to encrypt the file(s):

- AES128 has a 128-bit block size with a key size of 128 bits.
- AES192 has a 128-bit block size with a key size of 192 bits.
- AES256 has a 128-bit block size with a key size of 256 bits.

To use AES encryption for any database other than Oracle on a 32-bit platform, the path of the lib sub-directory of the Oracle GoldenGate installation directory must be specified as an environment variable before starting any processes. This is not required on 64-bit platforms. Set the path as follows:

- Linux: Specify the path as an entry to the LD_LIBRARY_PATHvariable. For example:

  ```
  setenv LD_LIBRARY_PATH ./lib:$LD_LIBRARY_PATH
  ```

- For Solaris: Specify the path as an entry to the SHLIB_PATH variable.
- For IBMi and AIX: Specify the path as an entry to the LIBPATH variable.
- For Windows: Add the path to the PATH variable.

You can use the SETENV parameter to set it as a session variable for the process.

NOENCRYPTTRAIL
Prevents the trail from being encrypted. This is the default.

Examples

Example 1
In the following example, the master key and wallet method is used. The Extract process writes to two trails. The data for the emp table is written to trail /home/ggsora/dirdat/em, which is encrypted with the AES-192 cipher. The data for the stores table is written to trail /home/ggsora/dirdat/st, which is not encrypted.

```
ENCRYPTTRAIL AES192
RMTTRAIL /home/ggsora/dirdat/em
TABLE hr.emp;
NOENCRYPTTRAIL
RMTTRAIL /home/ggsora/dirdat/st
TABLE ops.stores;
```

Example 2
As an alternative to the preceding example, you can omit NOENCRYPTTRAIL if you list all non-encrypted trails before the ENCRYPTTRAIL parameter.

```
RMTTRAIL /home/ggsora/dirdat/st
TABLE ops.stores;
ENCRYPTTRAIL AES192
RMTTRAIL /home/ggsora/dirdat/em
TABLE hr.emp;
```

Example 3
In the following example, the ENCKEYS method is used.

```
ENCRYPTTRAIL AES192, KEYNAME mykey1
RMTTRAIL /home/ggsora/dirdat/em
TABLE hr.emp;
TABLE ops.stores;
```

2.54 END

Valid For

Replicat

Description

Use the END parameter to terminate Replicat when it encounters the first record in the data source whose timestamp is the specified point in time.

Without END, the process runs continuously until:

- the end of the transaction log or trail is reached, at which point it will stop gracefully.
- manually terminated from the command shell.

Use END with the SPECIALRUN parameter to post data as a point-in-time snapshot, rather than continuously updating the target tables.

Default

Continuous processing

Syntax

```
END {date [time] | RUNTIME}
```

date [time]
Causes Replicat to terminate when it reaches a record in the data source whose timestamp exceeds the one that is specified with this parameter.
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 Replicat to terminate when it reaches a record in the data source whose timestamp exceeds the current date and clock time. All unprocessed records with

timestamps up to this point in time are processed. 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. Instead, you can control the process start time within your batch programming.

Examples

Example 1

```
SPECIALRUN
END 2010-12-31 17:00:00
```

Example 2

```
SPECIALRUN
END RUNTIME
```

2.55 EOFDELAY | EOFDELAYCSECS

Valid For

Extract and Replicat

Description

Use the `EOFDELAY` or `EOFDELAYCSECS` parameter to control how often Extract, a data pump, or Replicat checks for new data after it has reached the end of the current data in its data source. You can reduce the system I/O overhead of these reads by increasing the value of this parameter.

> **Note:**
>
> Large increases can increase the latency of the target data, especially when the activity on the source database is low

This parameter is not valid when `SOURCEISTABLE` is used. This parameter cannot be set to zero (0).

Default

The minimum is 1 second (1 second or 100 centiseconds ; the maximum is 60 seconds (60 seconds or 6000 centiseconds). The default is 1 second (100 centiseconds).

Syntax

```
EOFDELAY seconds | EOFDELAYCSECS centiseconds
```

seconds
The delay, in seconds, before searching for data to process.

centiseconds
The delay, in centiseconds, before searching for data to process.

Example

```
EOFDELAY 3
```

2.56 EXCLUDEHIDDENCOLUMNS

Valid For

Oracle Integrated Extract Capture; It's not valid for data pump.

Description

The parameter disables all the Oracle hidden columns including the timestamp columns created using automatic CDR. The parameter requires Oracle GoldenGate 12c (12.2.01) format trail or higher and must not specify the `NO_OBJECTDEFES` parameter. The `userexit` callback structure has the hidden column attributes and callback structure version is 5. You can specify the parameter at any location of the parameter file, as long as it is after the `EXTRACT group` parameter.

Syntax

```
EXTRACT ext1...
EXCLUDEHIDDENCOLUMNS
EXTTRAIL ./dirdat/a1
TABLE src.tab1;
```

2.57 EXCLUDETAG

Valid For

(Oracle) Extract and Replicat or data pump

(All databases) Extract Pump or Replicat

Description

Use `EXCLUDETAG` *`tag`* in your data pump or Replicat parameter file to specify changes to be excluded from trail files. The limitation for this parameter is that the tag value can be up to 2000 hexadecimal digits (0-9A-F) or the plus sign (`+`). You can have multiple `EXCLUDETAG` lines, but each `EXCLUDETAG` should have a single value. By default, Replicat the individual records every change it applies to the database by `00` in both classic mode or integrated mode.Compared with older versions, new trail file contains tag tokens, which would not introduce problems for older trail readers.

Use `EXCLUDETAG +` to ignore the individual records that are tagged with any redo tag.

Do not use `NULL` with `tag` or + because it operates in conflict resulting in errors.

To tag the individual records, use the `DBOPTIONS` parameter with the `SETTAG` option in the Replicat parameter file. Use these parameters to prevent cycling (loop-back) of Replicat the individual records in a bi-directional configuration or to filter other transactions from capture. The default `SETTAG` value is 00. Valid value is any single

Oracle Streams tag. A tag value can be up to 2000 hexadecimal digits (0-9 A-F) long. For more information about Streams tags, see *Oracle Streams Replication Administrator's Guide*.

Default

None

Syntax

```
[EXCLUDETAG [tag | NULL] | [+]
```

Example 1

For Replicat:

```
excludetag tag
```

Example 2

For data pumps:

```
excludetag 00
```

2.58 EXCLUDEWILDCARDOBJECTSONLY

Valid For

GLOBALS

Description

Use the `EXCLUDEWILDCARDOBJECTSONLY` parameter to force the inclusion of non-wildcarded source objects specified in `TABLE` or `MAP` parameters when an exclusion parameter contains a wildcard that otherwise would exclude that object. Exclusion parameters are `CATALOGEXCLUDE`, `SCHEMAEXCLUDE`, `MAPEXCLUDE`, and `TABLEEXCLUDE`.

The exclusion parameters get evaluated and satisfied before the `TABLE` or `MAP` statements. Without `EXCLUDEWILDCARDOBJECTSONLY`, it would be possible for an object in a `TABLE` or `MAP` statement to be wrongly excluded because it satisfies the wildcard in the exclude specification. For `EXCLUDEWILDCARDOBJECTSONLY` to work on an object, that object must be explicitly named without using wildcards in any of the name components.

Default

None

Syntax

```
EXCLUDEWILDCARDOBJECTSONLY
```

Example

In this example, `schema1.src_table1` is included in processing because the `TABLEEXCLUDE` parameter is wildcarded and the `TABLE` specification is not wildcarded. Without `EXCLUDEWILDCARDOBJECTSONLY`, `schema1.src_table1` would be excluded because of the wildcard specification in `TABLEEXCLUDE`.

```
TABLEEXCLUDE schema1.src_table*;
    TABLE schema1.src_table1;
```

2.59 EXTFILE

Valid For

Extract and Replicat

Description

Use the EXTFILE parameter to specify an extract file, which is a local file that will be read by a data pump Extract group on the local system, or to specify a local extract file that Replicat reads when SPECIALRUN is used.

Use this parameter for initial-load configurations. For online change synchronization, use the EXTTRAIL parameter. The size of an Extract file cannot exceed 2GB.

EXTFILE must precede all associated TABLE or MAP statements. Multiple EXTFILE statements can be used to define different files.

EXTFILE parameter is deprecated and ignored for Data Pump

You can encrypt the data in this file by using the ENCRYPTTRAIL parameter. See "ENCRYPTTRAIL | NOENCRYPTTRAIL" for more information.

Default

None

Syntax

```
EXTFILE file_name
[, FORMAT RELEASE major.minor]
[, MEGABYTES megabytes]
[,     OBJECTDEFS | NO_OBJECTDEFS]
[, TRAILBYTEORDER {BIGENDIAN | LITTLEENDIAN | NATIVEENDIAN}]
```

file_name
Valid for Extract and Replicat. Specifies the relative or fully qualified name of the extract file.

FORMAT RELEASE *major.minor*
Specifies the metadata format of the data that is sent by Extract to a trail, a file, or (if a remote task) to another process. The metadata tells the reader process whether the data records are of a version that it supports. The metadata format depends on the version of the Oracle GoldenGate process. Older Oracle GoldenGate versions contain different metadata than newer ones.

- FORMAT is a required keyword.
- RELEASE specifies an Oracle GoldenGate release version. *major* is the major version number, and *minor* is the minor version number. The X.x must reflect a current or earlier, generally available (GA) release of Oracle GoldenGate. Valid values are 9.0 through the current Oracle GoldenGate X.x version number, for example 11.2 or 12.1. (If you use an Oracle GoldenGate version that is earlier

than 9.0, specify either 9.0 or 9.5.) The release version is programmatically mapped back to the appropriate trail format compatibility level. The default is the current version of the process that writes to this trail. Note that RELEASE versions earlier than 12.1 do not support three-part object names.

The following settings are supported in Oracle Database 12.2:

- For Oracle Database 12.2 non-CDB with compatibility set to 12.1, FORMAT RELEASE 12.2 or above is supported.
- For Oracle Database 12.2 non-CDB with compatibility set to 12.2, FORMAT RELEASE 12.2 or above is supported.
- For Oracle Database 12.2 CDB/PDB with compatibility set to 12.2, only FORMAT RELEASE 12.3 is supported. This is due to the use of local undo for PDBs, which requires augmenting the transaction ID with the PDB number to ensure uniqueness of trx IDs.

MEGABYTES *megabytes*
Valid for Extract. The maximum size, in megabytes, of a file in the trail. The default is 2000.

OBJECTDEFS | NO_OBJECTDEFS
Use the OBJECTDEFS and NO_OBJECTDEFS options to control whether or not to include the object definitions in the trail. These two options are applicable only when the output trail is formatted in Oracle GoldenGate canonical format and the trail format release is greater than 12.1. Otherwise, both options are ignored because no metadata record will be added to the trail.
When replicating from an Open Systems database to NonStop, specify format version below 12.2 to avoid including the object definitions in the trail since NonStop does not support processing object definitions from the trail.

TRAILBYTEORDER {BIGENDIAN | LITTLEENDIAN | NATIVEENDIAN}
Sets the byte format of the metadata in the file records. This parameter does not affect the column data. Valid only for files that have a FORMAT RELEASE version of at least 12.1. Valid values are BIGENDIAN (big endian), LITTLEENDIAN (little endian), and NATIVEENDIAN (default of the local system). The default is BIGENDIAN. See the GLOBALS version of TRAILBYTEORDER for additional usage instructions.

Examples

Example 1

```
EXTFILE dirdat/datafile
```

Example 2

```
EXTFILE dirdat/extdat, MEGABYTES 2
```

Example 3

```
EXTFILE /ggs/dirdat/extdat, FORMAT RELEASE 10.4
```

2.60 EXTRACT

Valid For

Extract

Description

Use the EXTRACT parameter to specify an Extract group for online (continuous) change synchronization. This parameter links the current run with previous runs, so that data continuity is maintained between source and target tables. Unless stopped by a user, Extract runs continuously and maintains checkpoints in the data source and trail to ensure data integrity and fault tolerance throughout planned or unplanned process termination, system outages, or network failure. EXTRACT must be the first entry in the parameter file.

For more information about implementing change synchronization, see the *Administering Oracle GoldenGate*.

Default

None

Syntax

```
EXTRACT group_name
```

group_name
The group name as defined with the ADD EXTRACT command.

Example

The following specifies an Extract group named finance.

```
EXTRACT finance
```

2.61 EXTTRAIL

Valid For

Extract

Description

Use the EXTTRAIL parameter to specify a trail on the local system that was created with the ADD EXTTRAIL command. The trail is read by a data pump Extract group or by a Replicat group on the local system.

EXTTRAIL must precede all associated TABLE statements. Multiple EXTTRAIL statements can be used to define different trails.

Do not use EXTTRAIL for an Extract that is configured in PASSIVE mode. See the *Administering Oracle GoldenGate* guide for more information about PASSIVE mode, an Oracle GoldenGate security feature.

EXTTRAIL parameter is deprecated and ignored for Data Pump.

You can encrypt the data in this trail by using the ENCRYPTTRAIL parameter. See "ENCRYPTTRAIL | NOENCRYPTTRAIL" for more information.

Default

None

Syntax

```
EXTTRAIL file_name
[, FORMAT RELEASE major.minor]
[, OBJECTDEFS | NO_OBJECTDEFS]
[, TRAILBYTEORDER {BIGENDIAN | LITTLEENDIAN | NATIVEENDIAN}]
```

file_name
The relative or fully qualified name of the trail. Use a maximum of two characters for the name. As trail files are aged, a nine-character sequence number will be added to this name, for example `/ogg/dirdat/rt000000001`.

`FORMAT RELEASE` ***`major.minor`***
Specifies the metadata format of the data that is sent by Extract to a trail, a file, or (if a remote task) to another process. The metadata tells the reader process whether the data records are of a version that it supports. The metadata format depends on the version of the Oracle GoldenGate process. Older Oracle GoldenGate versions contain different metadata than newer ones.

- `FORMAT` is a required keyword.
- `RELEASE` specifies an Oracle GoldenGate release version. *`major`* is the major version number, and *`minor`* is the minor version number. The `X.x` must reflect a current or earlier, generally available (GA) release of Oracle GoldenGate. Valid values are 9.0 through the current Oracle GoldenGate `X.x` version number.

  For the Oracle GoldenGate 18c (18.1.0.0) release, the valid value is 18.1. If you use an Oracle GoldenGate version that is earlier than 9.0, specify either 9.0 or 9.5.

  The release version is programmatically mapped back to an appropriate internal compatibility level. The default is the current version of the process that writes to this trail.

  > **Note:**
  >
  > `RELEASE` versions earlier than 12.1 do not support three-part object names.

There is a dependency between `FORMAT` and the `RECOVERYOPTIONS` parameter. See "RECOVERYOPTIONS" for more information.
The following settings are supported in Oracle Database 12.2:

- For Oracle Database 12.2 non-CDB with compatibility set to 12.1, `FORMAT RELEASE` 12.2 or above is supported.
- For Oracle Database 12.2 non-CDB with compatibility set to 12.2, `FORMAT RELEASE` 12.2 or above is supported.
- For Oracle Database 12.2 CDB/PDB with compatibility set to 12.2, `FORMAT RELEASE` values `12.3` and `18.1` are supported. This is due to the use of local undo for PDBs, which requires augmenting the transaction ID with the PDB number to ensure uniqueness of trx IDs.

OBJECTDEFS | NO_OBJECTDEFS
Use the OBJECTDEFS and NO_OBJECTDEFS options to control whether or not to include the object definitions in the trail. These two options are applicable only when the output trail is formatted in Oracle GoldenGate canonical format and the trail format release is greater than 12.1. Otherwise, both options are ignored because no metadata record will be added to the trail.
When replicating from an Open Systems database to NonStop, specify format version below 12.2 to avoid including the object definitions in the trail since NonStop does not support processing object definitions from the trail.

TRAILBYTEORDER {BIGENDIAN | LITTLEENDIAN | NATIVEENDIAN}
Sets the byte format of the metadata in the trail records. This parameter does not affect the column data. Valid only for trails that have a FORMAT RELEASE version of at least 12.1. Valid values are BIGENDIAN (big endian), LITTLEENDIAN (little endian), and NATIVEENDIAN (default of the local system). The default is BIGENDIAN. See the GLOBALS version of TRAILBYTEORDER for additional usage instructions.

Examples

Example 1

```
EXTTRAIL dirdat/ny
```

Example 2

```
EXTTRAIL /ggs/dirdat/ex, FORMAT RELEASE 10.4
```

2.62 FETCHOPTIONS

Valid For

Extract

Description

Use the FETCHOPTIONS parameter to control certain aspects of the way that Oracle GoldenGate fetches data in the following circumstances:

- When the transaction record does not contain enough information for Extract to reconstruct an update operation.
- When Oracle GoldenGate must fetch a column value as the result of a FETCHCOLS clause of a TABLE statement.

FETCHOPTIONS is table-specific. One FETCHOPTIONS statement applies for all subsequent TABLE statements until a different FETCHOPTIONS statement is encountered.

Default fetch properties are adequate for most installations.

Default

Ignore missing rows and continue processing

Syntax

```
FETCHOPTIONS
[, FETCHPKUPDATECOLS]
[, INCONSISTENTROW action]
[, MAXFETCHSTATEMENTS number]
```

```
[, MISSINGROW action]
[, NOFETCH]
[, SUPPRESSDUPLICATES]
[, USEKEY | NOUSEKEY]
[, USELATESTVERSION | NOUSELATESTVERSION]
[, USESNAPSHOT | NOUSESNAPSHOT]
[, USEROWID | NOUSEROWID]
```

FETCHPKUPDATECOLS
Fetches all unavailable columns when a primary key is updated. This option is off by default. When off, column fetching is performed according to other FETCHOPTIONS options that are enabled.
When on, it only takes effect during an update to a primary key column. The results are the same as using FETCHCOLS (*) in the TABLE statement. LOB columns are included in the fetch.
Use this parameter when using HANDLECOLLISIONS. When Replicat detects a missing update, all of the columns will be available to turn the update into an insert.

INCONSISTENTROW *action*
Indicates that column data was successfully fetched by row ID, but the key did not match. Either the row ID was recycled or a primary key update occurred after this operation (and prior to the fetch).
action can be one of the following:

ALLOW
Allow the condition and continue processing.

IGNORE
Ignore the condition and continue processing. This is the default.

REPORT
Report the condition and contents of the row to the discard file, but continue processing the partial row.

DISCARD
Discard the data and do not process the partial row.

ABEND
Discard the data and quit processing.

MAXFETCHSTATEMENTS *number*
Controls the maximum allowable number of prepared queries that can be used by Extract to fetch row data from a source database. The fetched data is used when not enough information is available to construct a logical SQL statement from a transaction log record. Queries are prepared and cached as needed. When the value set with MAXFETCHSTATEMENTS is reached, the oldest query is replaced by the newest one. The value of this parameter controls the number of open cursors maintained by Extract for fetch queries only. Additional cursors may be used by Extract for other purposes, such as those required for stored procedures. This parameter is only valid for Oracle databases.The default is 100 statements. Make certain that the database can support the number of cursors specified, plus cursors used by other applications and processes.

MISSINGROW *action*
Provides a response when Oracle GoldenGate cannot locate a row to be fetched, causing only part of the row (the changed values) to be available for processing.

Typically a row cannot be located because it was deleted between the time the change record was created and when the fetch was triggered, or because the row image required was older than the undo retention specification.
action can be one of the following:

ALLOW
Allow the condition and continue processing. This is the default.

IGNORE
Ignore the condition and continue processing.

REPORT
Report the condition and contents of the row to the discard file, but continue processing the partial row.

DISCARD
Discard the data and do not process the partial row.

ABEND
Discard the data and quit processing.

NOFETCH
Prevents Extract from fetching the column from the database, this option is off by default. Extract writes the record to the trail, but inserts a token indicating that the column is missing.

SUPPRESSDUPLICATES
Valid for Oracle. Avoids target tablespaces becoming overly large when updates are made on LOB columns. By default, `SUPPRESSDUPLICATES` is set to off. For example, after replication a source tablespace of 232MB becomes a target tablespace of 7.52GB.

USEKEY | NOUSEKEY
Determines whether or not Oracle GoldenGate uses the primary key to locate the row to be fetched.
If both `USEKEY` and `USEROWID` are specified, `ROWID` takes priority for faster access to the record. `USEROWID` is the default.

USELATESTVERSION | NOUSELATESTVERSION
Valid for Oracle. Use with `USESNAPSHOT`. The default, `USELATESTVERSION`, directs Extract to fetch data from the source table if it cannot fetch from the undo tablespace. `NOUSELATESTVERSION` directs Extract to ignore the condition if the snapshot fetch fails, and continue processing.
To provide an alternate action if a snapshot fetch does not succeed, use the `MISSINGROW` option.

USESNAPSHOT | NOUSESNAPSHOT
Valid for Oracle. The default, `USESNAPSHOT`, causes Extract to use the Oracle Flashback mechanism to fetch the correct snapshot of data that is needed to reconstruct certain operations that cannot be fully captured from the redo record. `NOUSESNAPSHOT` causes Extract to fetch the needed data from the source table instead of the flashback logs.

USEROWID | NOUSEROWID
Valid for Oracle. Determines whether or not Oracle GoldenGate uses the row ID to locate the row to be fetched.

If both USEKEY and USEROWID are specified, ROWID takes priority for faster access to the record. USEROWID is the default.

Examples

Example 1
The following directs Extract to fetch data by using Flashback Query and to ignore the condition and continue processing the record if the fetch fails.

```
FETCHOPTIONS USESNAPSHOT, NOUSELATESTVERSION
```

Example 2

```
MAXFETCHSTATEMENTS 150
```

Example 3
The following directs Extract to fetch data by using Flashback Query and causes Extract to abend if the data is not available.

```
FETCHOPTIONS USESNAPSHOT, NOUSELATESTVERSION, MISSINGROW ABEND
```

2.63 FETCHUSERID

Valid For

Integrated primary Extract on Oracle; data pump Extract is not valid

Description

Use the FETCHUSERID parameter to specify the type of authentication for an Oracle GoldenGate process to use when logging into a database, and to specify password encryption information. This parameter can be used instead of FETCHUSERIDALIAS when an Oracle GoldenGate credential store is not being used.

Always use FETCHUSERID or FETCHUSERIDALIAS for a primary Extract. Use FETCHUSERID or FETCHUSERIDALIAS for a data pump Extract.

FETCHUSERID Compared to FETCHUSERIDALIAS

FETCHUSERID requires either specifying the clear-text password in the parameter file or encrypting it with the ENCRYPT PASSWORD command and, optionally, storing an encryption key in an ENCKEYS file. FETCHUSERID supports a broad range of the databases that Oracle GoldenGate supports.

FETCHUSERIDALIAS enables you to specify an alias, rather than a user ID and password, in the parameter file. The user IDs and encrypted passwords are stored in a credential store. FETCHUSERIDALIAS supports databases running on Linux, UNIX, and Windows platforms.

FETCHUSERID Requirements

FETCHUSERID is not always required, nor is PASSWORD always required when FETCHUSERID is required. In some cases, it is sufficient just to use FETCHUSERID or even just to use the SOURCEDB or TARGETDB parameter, depending on how authentication for the database is configured.

See "SOURCEDB" and "TARGETDB" for more information.

Use FETCHUSERID for Oracle GoldenGate processes that connect to an Oracle database. The purpose of this connection is to offload fetch operations to an Active Data Guard standby database, which eliminates overhead that would otherwise be placed on the source database.

- To use an operating system login, use FETCHUSERID with the / argument.
- To use a database user name and password, use FETCHUSERID with PASSWORD.
- Optionally, you can specify the user to log in as sysdba.
- (*Oracle Enterprise Edition earlier than 11.2.0.2*) Special database privileges are required for the FETCHUSERID user when Extract is configured to use LOGRETENTION. These privileges might have been granted when Oracle GoldenGate was installed. See the Ensuring Data Availability for Classic Capture in *Using Oracle GoldenGate for Oracle Database*for more information about LOGRETENTION.
- (*Oracle Standard or Enterprise Edition 11.2.0.2 or later*) To use FETCHUSERID for an Extract group that is configured for integrated capture, the user must have the privileges granted in the dbms_goldengate_auth.grant_admin_privilege.
- To support capture from an Oracle container database, the user that is specified with FETCHUSERID must log into the root container and must be a common user. A connect string must be supplied for this user and must include the required C## prefix of the common user, such as C##GGADMIN@FINANCE. For more information, see Assigning Credentials to Oracle GoldenGate in *Using Oracle GoldenGate for Oracle Database*.
- The connection specified by FETCHUSERI or FETCHUSERIDALIAS must be to an Active Data Guard standby database of the source database.
- FETCHUSERID can be specified anywhere in the parameter file. Ordering does not matter. It can come before or after a TABLE or MAP statement.

Default

None

Syntax

```
FETCHUSERID {/ | user}[, PASSWORD password]
[algorithm ENCRYPTKEY {key_name | DEFAULT}] [SYSDBA]
```

/
Directs Oracle GoldenGate to use an operating-system login for Oracle, not a database user login. Use this argument only if the database allows authentication at the operating-system level. Bypassing database-level authentication eliminates the need to update Oracle GoldenGate parameter files if application passwords frequently change. To use this option, the correct user name must exist in the database, in relation to the value of the Oracle OS_AUTHENT_PREFIX initialization parameter, as follows:

- The value set with OS_AUTHENT_PREFIX is concatenated to the beginning of a user's operating system account name and then compared to the database name. Those two names must match.

- If OS_AUTHENT_PREFIX is set to ' ' (a null string), the user name must be created with IDENTIFIED EXTERNALLY. For example, if the OS user name is ogg, you would use the following to create the database user:

  ```
  CREATE USER ogg IDENTIFIED EXTERNALLY;
  ```

- If OS_AUTHENT_PREFIX is set to OPS$ or another string, the user name must be created in the following format:

  OS_AUTHENT_PREFIX_value OS_user_name

  For example, if the OS user name is ogg, you would use the following to create the database user:

  ```
  CREATE USER ops$ogg IDENTIFIED BY oggpassword;
  ```

user
Specifies the name of a database user or a schema, depending on the database configuration. A SQL*Net connect string can be used.

password
Use when database authentication is required to specify the password for the database user. If the password was encrypted by means of the ENCRYPT PASSWORD command, supply the encrypted password; otherwise, use the clear-text password. If the password is case-sensitive, type it that way.
If either the user ID or password changes, the change must be made in the Oracle GoldenGate parameter files, including the re-encryption of the password if necessary.

algorithm
Specifies the encryption algorithm that was used to encrypt the password with ENCRYPT PASSWORD.
The algorithm can be one of:
AES128
AES192
AES256
BLOWFISH

ENCRYPTKEY {*key_name* | DEFAULT}
Specifies the encryption key that was specified with ENCRYPT PASSWORD.

- ENCRYPTKEY *key_name* specifies the logical name of a user-created encryption key in the ENCKEYS lookup file. Use if ENCRYPT PASSWORD was used with the *KEYNAME key_name* option.
- ENCRYPTKEY DEFAULT directs Oracle GoldenGate to use a random key. Use if ENCRYPT PASSWORD was used with the KEYNAME DEFAULT option.

SYSDBA
Specifies that the user logs in as sysdba.

Example

```
fetchuserid gg_user@adg_inst password pwd
```

2.64 FETCHUSERIDALIAS

Valid For

Integrated primary Extract on Oracle; data pump Extract is not valid

Description

Use the FETCHUSERIDALIAS parameter to specify authentication for an Oracle GoldenGate process to use when logging into a database. The use of FETCHUSERIDALIAS requires the use of an Oracle GoldenGate credential store. Specify FETCHUSERIDALIAS before any TABLE or MAP entries in the parameter file.

FETCHUSERIDALIAS Compared to FETCHUSERID

FETCHUSERIDALIAS enables you to specify an alias, rather than a user ID and password, in the parameter file. The user IDs and encrypted passwords are stored in a credential store. FETCHUSERIDALIAS supports databases running on Linux, UNIX, and Windows platforms.

FETCHUSERID requires either specifying the clear-text password in the parameter file or encrypting it with the ENCRYPT PASSWORD command and, optionally, storing an encryption key in an ENCKEYS file. FETCHUSERID supports a broad range of the databases that Oracle GoldenGate supports. In addition, it supports the use of an operating system login for Oracle databases.

FETCHUSERID Requirements

> **Note:**
>
> Logins that require a database user and password must be stored in the Oracle GoldenGate credential store.

Use FETCHUSERIDALIAS for Oracle GoldenGate processes that connect to an Oracle database. The purpose of this connection is to offload fetch operations to an Active Data Guard standby database, which eliminates overhead that would otherwise be placed on the source database.

- The SOURCEDB or TARGETDB parameter is not required.
- Specify the alias of a database credential that is stored in the Oracle GoldenGate credential store.
- (*Oracle Enterprise Edition earlier than 11.2.0.2*) Special database privileges are required for the FETCHUSERIDALIAS user when Extract is configured to use LOGRETENTION. These privileges might have been granted when Oracle GoldenGate was installed. See the Ensuring Data Availability for Classic Capture in*Using Oracle GoldenGate for Oracle Database* for more information about LOGRETENTION.
- (*Oracle Standard or Enterprise Edition 11.2.0.2 or later*) To use FETCHUSERIDALIAS for an Extract group that is configured for integrated capture, the user must have the privileges granted in the dbms_goldengate_auth.grant_admin_privilege.

- To support capture from an Oracle container database, the user that is specified with FETCHUSERID must log on to the root container and must be a common database user. A connect string must be supplied for this user, for example: C##GGADM@FINANCE. For more information, see Establishing Oracle GoldenGate Credentials in *Using Oracle GoldenGate for Oracle Database*.
- The connection specified by FETCHUSERI or FETCHUSERIDALIAS must be to an Active Data Guard standby database of the source database.
- FETCHUSERID can be specified anywhere in the parameter file. Ordering does not matter. It can come before or after a TABLE or MAP statement.

Default

None

Syntax

```
FETCHUSERIDALIAS alias [DOMAIN domain] [SYSDBA]
```

alias
Specifies the alias of a database user credential that is stored in the Oracle GoldenGate credential store.

DOMAIN *domain*
Specifies the credential store domain for the specified alias. A valid domain entry must exist in the credential store for the specified alias.

SYSDBA
Specifies that the user logs in as sysdba.

Example

```
fetchuseridalias gg_user@adg_inst password pwd
```

2.65 FILTERDUPS | NOFILTERDUPS

Valid For

Replicat

Description

Use the FILTERDUPS and NOFILTERDUPS parameters to handle anomalies that can occur on a NonStop system when an application performs multiple operations on the same record within the same transaction. This type of transaction can cause out-of-order records in the TMF audit trail and will cause Replicat to abend. For example:

- An insert can occur in the audit trail before a delete on the same primary key, even though the source application performed the delete first, followed by the insert (resulting in a duplicate-record error when the insert is performed by Replicat).
- An update can occur in the audit trail before an insert on the same primary key (resulting in a missing-record error when the update is performed by Replicat).

FILTERDUPS prevents Replicat from abending by resolving the conditions as follows:

- In the event of a duplicate insert, Replicat saves the duplicated insert until the end of the transaction. If a delete with the same primary key is subsequently encountered, Replicat performs the delete, then the insert.
- In the event of a missing update, Replicat saves the missing update until the end of the transaction. If an insert with the same primary key is subsequently encountered, Replicat performs the insert, then the update.

IDX hospital applications and some BASE24 bank applications are the typical, but not the only, sources of this anomaly. Use `FILTERDUPS` only if Replicat is abending on duplicate or missing records and you know they were caused by out-of-order transactions originating on a NonStop system. The Logdump utility can be used to diagnose this condition.

`FILTERDUPS` and `NOFILTERDUPS` can be used as on-off switches for different groups of `MAP` statements to enable or disable the exception processing as needed.

Default

`NOFILTERDUPS`

Syntax

```
FILTERDUPS | NOFILTERDUPS
```

Example

This example turns on `FILTERDUPS` for `ORDERS` but disables it for any `MAP` statements that are defined later in the same parameter file.

```
FILTERDUPS
MAP $DATA1.SQLDAT.ORDERS, TARGET MASTER.ORDERS;
NOFILTERDUPS
```

2.66 FLUSHSECS | FLUSHCSECS

Valid For

Extract

Description

Use the `FLUSHSECS` or `FLUSHCSECS` parameters to control when Oracle GoldenGate flushes the Extract memory buffer. When sending data to remote systems, Extract buffers data to optimize network performance. The buffer is flushed to the target system when it is full or after the amount of time specified with `FLUSHSECS` or `FLUSHCSECS`. Data changes are not available to the target users until the buffer is flushed and the data is posted. To control the size of the buffer, use the `TCPBUFSIZE` option of `RMTHOST`. See "RMTHOST" for more information.

Increasing the value of `FLUSHSECS` or `FLUSHCSECS` could result in slightly more efficient use of the network, but it could increase the latency of the target data if activity on the source system is low and the buffer does not fill up. When source tables remain busy, `FLUSHSECS` and `FLUSHCSECS` have little effect.

This parameter cannot be set to zero (0).

Default

The default is 1. The minimum is 0; the maximum is 5000.

Syntax

```
FLUSHSECS seconds | FLUSHCSECS centiseconds
```

seconds
The delay, in seconds, before flushing the buffer.

centiseconds
The delay, in centiseconds, before flushing the buffer.

Example

```
FLUSHSECS 80
```

2.67 FUNCTIONSTACKSIZE

Valid For

Extract and Replicat

Description

Use the FUNCTIONSTACKSIZE parameter to control the size of the memory stack that is used for processing Oracle GoldenGate column-conversion functions. The memory stack holds arguments supplied to and from an Oracle GoldenGate function. You should not need to use this parameter unless Oracle GoldenGate returns a message indicating that the size of the stack should be increased. The message is similar to:

```
Not enough stack space. Specify FUNCTIONSTACKSIZE greater than {0,number,0}
```

This could happen when you are using a very large number of functions or arguments.

The default without FUNCTIONSTACKSIZE is 200 arguments, which optimizes the performance of Oracle GoldenGate and its usage of system memory. Increasing this parameter can adversely affect performance and the use of system memory.

When setting FUNCTIONSTACKSIZE for a coordinated Replicat, take into account that the specified value is applied to each thread in the configuration, not as an aggregate threshold for Replicat as a whole. For example, if FUNCTIONSTACKSIZE 400 is specified, it is possible for each thread to have 399 arguments without any warning or error from Replicat.

FUNCTIONSTACKSIZE must appear in the parameter file before any parameters that include functions are listed. FUNCTIONSTACKSIZE is a global parameter. It affects all clauses in a parameter file.

Default

200 arguments

Syntax

```
FUNCTIONSTACKSIZE number
```

number
A value between 0 and 5000 that denotes the number of function arguments to allow in a parameter clause.

Example

```
FUNCTIONSTACKSIZE 300
```

2.68 GETAPPLOPS | IGNOREAPPLOPS

Valid For

Extract

Description

Use the GETAPPLOPS or IGNOREAPPLOPS parameter to capture or ignore DML operations produced by any application except the local Replicat. By default, application data is captured.

These parameters are useful in conjunction with the GETREPLICATES and IGNOREREPLICATES parameters for the following:

- To separate data operations performed by a local Replicat from those performed by the business applications configured for Oracle GoldenGate extraction. Use IGNOREAPPLOPS and GETREPLICATES for one trail or file to contain just the Replicat operations, and use GETAPPLOPS and IGNOREREPLICATES for another trail or file to contain just the operations of the business applications.
- As part of a cascading configuration, where changes applied by Replicat locally must be captured by a local Extract to be propagated to another system. In this case, IGNOREAPPLOPS and GETREPLICATES would be used.
- As part of a loop detection scheme when using bidirectional replication. The default combination of GETAPPLOPS and IGNOREREPLICATES causes Extract to capture application data while ignoring Replicat operations posted to the same database objects. In addition to using these parameters, Extract must be configured to identify Replicat transactions.

See "GETREPLICATES | IGNOREREPLICATES" for more information.

Using GETAPPLOPS for Oracle Sequences

GETAPPLOPS must be enabled to capture sequences that are replicated by Replicat. Replicat issues sequence updates in an autonomous transaction, so they are not reflected in the trace table. The sequence update appears as if it is an application operation.

Using GETAPPLEOPS for DDL Operations

See "DDLOPTIONS" for information on to use GETAPPLOPS or IGNOREAPPLOPS functionality for DDL operations.

For more information about configuring bidirectional replication, see the *Administering Oracle GoldenGate*.

Default

GETAPPLOPS

Syntax

GETAPPLOPS | IGNOREAPPLOPS

2.69 GETDELETES | IGNOREDELETES

Valid For

Extract and Replicat

Description

Use the GETDELETES and IGNOREDELETES parameters to control whether or not Oracle GoldenGate processes DELETE operations. These parameters are table-specific. One parameter remains in effect for all subsequent TABLE or MAP statements, until the other parameter is encountered.

Because you can selectively enable or disable these parameters between MAP statements, you can enable or disable them for different threads of a coordinated Replicat. Specify the GETDELETES threads in one set of MAP statements, and specify the IGNOREDELETES threads in a different set of MAP statements.

Default

GETDELETES

Syntax

GETDELETES | IGNOREDELETES

Example

This example shows how you can apply GETDELETES and IGNOREDELETES selectively to different MAP statements, each of which represents a different thread of a coordinated Replicat.

```
GETDELETES
MAP sales.cust, TARGET sales.cust, THREAD (1);
MAP sales.ord, TARGET sales.ord, THREAD (2);
IGNOREDELETES
MAP sales.loc, TARGET sales.loc, THREAD (3);
```

2.70 GETINSERTS | IGNOREINSERTS

Valid For

Extract and Replicat

Description

Use the GETINSERTS and IGNOREINSERTS parameters to control whether or not INSERT operations are processed by Oracle GoldenGate. These parameters are table-specific.

One parameter remains in effect for all subsequent TABLE or MAP statements, until the other parameter is encountered.

Because you can selectively enable or disable these parameters between MAP statements, you can enable or disable them for different threads of a coordinated Replicat. Specify the GETINSERTS threads in one set of MAP statements, and specify the IGNOREINSERTS threads in a different set of MAP statements.

Default

```
GETINSERTS
```

Syntax

```
GETINSERTS | IGNOREINSERTS
```

Example

This example shows how you can apply GETINSERTS and IGNOREINSERTS selectively to different MAP statements, each of which represents a different thread of a coordinated Replicat.

```
GETINSERTS
MAP sales.cust, TARGET sales.cust, THREAD (1);
MAP sales.ord, TARGET sales.ord, THREAD (2);
IGNOREINSERTS
MAP sales.loc, TARGET sales.loc, THREAD (3);
```

2.71 GETREPLICATES | IGNOREREPLICATES

Valid For

Extract

Description

Use the GETREPLICATES and IGNOREREPLICATES parameters to control whether or not DML transactions issued by Replicat are captured or ignored by an Extract process that is processing the same tables on the same system.

Ignoring Replicat Transactions

By default, Extract uses a combination of IGNOREREPLICATES and GETAPPLOPS . In this configuration, Extract captures all application data that is configured for synchronization by Oracle GoldenGate, and it ignores all Replicat operations. In a bi-directional configuration, this prevents the data that Replicat applies from looping back to the original system, which would cause duplicate-record errors.

Capturing Replicat Transactions

Use GETREPLICATES with IGNOREAPPLOPS in a cascading configuration to enable replicated data to be captured again by Extract on an intermediary system so that it can be replicated to the final target. For example, if database A replicates to database B, and database B replicates to database C, you would use GETREPLICATES for the Extract on database B.

> **Note:**
>
> Even with GETREPLICATES in effect, however, you still can exclude specific replicated data from being captured by using a WHERE or FILTER clause in a TABLE or MAP statement.

Using GETREPLICATES and IGNOREREPLICATES with Oracle

The GETREPLICATES and IGNOREREPLICATES parameters should not be used if you are not using a trace table (the TRACETABLE parameter). By default, Extract captures *all* transactions including transactions committed by Replicat. If you want to ignore the Replicat transactions, you should use the TRANLOGOPTIONS EXCLUDEUSER parameter. You can also use this to ignore transactions by any specific user in addition to Replicat's user. TRANLOGOPTIONS EXCLUDETAG + is a better method than using TRANLOGOPTIONS EXCLUDEUSER.

If you are using the TRACETABLE parameter or have the default trace table, GGS_TRACE, created without explicitly using TRACETABLE, then Extract automatically ignores any transaction that has a TRACETABLE update in it by default. If you want to capture the Replicat committed transactions, you have to specify GETREPLICATES. In this case, Oracle does not recommend that you use it with TRANLOGOPTIONS EXCLUDUSER because Replicat will have unpredictable behavior in transaction filtering.

Default

```
IGNOREREPLICATES
```

Syntax

```
GETREPLICATES | IGNOREREPLICATES
```

2.72 GETTRUNCATES | IGNORETRUNCATES

Valid For

Extract and Replicat

Description

Use the GETTRUNCATES and IGNORETRUNCATES parameters to control whether or not Oracle GoldenGate processes table truncate operations. By default, truncate operations are not captured from the source or replicated to the target.

GETTRUNCATES and IGNORETRUNCATES are table-specific. One parameter remains in effect for all subsequent TABLE or MAP statements, until the other parameter is encountered.

In a coordinated Replicat configuration, truncates are always processed by the thread that is responsible for barrier transactions.

Supported Databases

- GETTRUNCATES and IGNORETRUNCATES are not supported for Teradata.

- GETTRUNCATES and IGNORETRUNCATES are supported by Extract for Oracle Database, MySQL, DB2 LUW, and DB2 for i.
- GETTRUNCATES and IGNORETRUNCATES are supported by Extract and Replicat for DB2 for i.
- GETTRUNCATES and IGNORETRUNCATES are supported by Replicat for Oracle Database, SQL Server, DB2 LUW, DB2 z/OS, MySQL, and other ODBC targets that support the TRUNCATE command.
- GETTRUNCATES and IGNORETRUNCATES are supported by Extract for Sybase. However, if the same table name is present in the include list for two schemas then it is abended and an error occurs, stating " Found the same table name in multiple schemas and the table name could not be resolved to process truncate operation."
- Sybase AES 16 is supported from Oracle GoldenGate 12.2.0.2.0

> **Note:**
>
> It is not possible to ignore TRUNCATEs during capture from a DB2 z/OS database. By default, TRUNCATEs are always captured from a DB2 z/OS source, but they can be ignored by Replicat if IGNORETRUNCATES is used in the Replicat parameter file.

DB2 LUW Limitations

- DB2 LUW does not support a TRUNCATE command, so Replicat replicates a truncate operation by performing an IMPORT REPLACE from a NULL (blank) file.

Oracle Limitations

- Oracle GoldenGate supports the Oracle TRUNCATE TABLE command, but not TRUNCATE PARTITION. You can replicate TRUNCATE PARTITION as part of the full Oracle GoldenGate DDL replication support.
- The database does not log truncates against an empty table, so those operations are not captured by Oracle GoldenGate. The DDL support of Oracle GoldenGate can be used for this purpose.
- The database does not log truncates for empty partitions, so Oracle GoldenGate cannot reliably process TRUNCATE TABLE when the table contains any empty partitions. Do not use GETTRUNCATES on any partitioned table. Oracle GoldenGate DDL support can be used to capture truncates on tables that might include empty partitions.
-

Default

```
IGNORETRUNCATES
```

Syntax

```
GETTRUNCATES | IGNORETRUNCATES
```

2.73 GETUPDATEAFTERS | IGNOREUPDATEAFTERS

Valid For

Extract and Replicat

Description

Use the GETUPDATEAFTERS and IGNOREUPDATEAFTERS parameters to control whether or not the after images of columns in UPDATE operations are included in the records processed by Oracle GoldenGate. After images contain the results of the UPDATE.

These parameters are table-specific. One parameter remains in effect for all subsequent TABLE or MAP statements, until the other parameter is encountered.

Because you can selectively enable or disable these parameters between MAP statements, you can enable or disable them for different threads of a coordinated Replicat. Specify the GETUPDATEAFTERS threads in one set of MAP statements, and specify the IGNOREUPDATEAFTERS threads in a different set of MAP statements.

Default

```
GETUPDATEAFTERS
```

Syntax

```
GETUPDATEAFTERS | IGNOREUPDATEAFTERS
```

Example

This example shows how you can apply GETUPDATEAFTERS and IGNOREUPDATEAFTERS selectively to different MAP statements, each of which represents a different thread of a coordinated Replicat.

```
GETUPDATEAFTERS
MAP sales.cust, TARGET sales.cust, THREAD (1);
MAP sales.ord, TARGET sales.ord, THREAD (2);
IGNOREUPDATEAFTERS
MAP sales.loc, TARGET sales.loc, THREAD (3);
```

2.74 GETUPDATEBEFORES | IGNOREUPDATEBEFORES

Valid For

Extract and Replicat

Description

Use the GETUPDATEBEFORES and IGNOREUPDATEBEFORES parameters to control whether or not the before images of columns in UPDATE operations are included in the records that are processed by Oracle GoldenGate. Before images contain column details that existed before a row was updated.

Oracle GoldenGate 12c captures both the pre-change and post-change values for update operations in a single unified update record by default. In previous releases the default was to only capture the post-change value. Beginning in this release, custom

SQL statements (SQLEXEC) now only execute once per update operation with the new default update format. Prior to this release, custom SQL statements would execute twice, once when encountering the pre-change value and once when encountering the post-change value. If you are using the Oracle GoldenGate 12c (12.1.*x* or 12.2.*x*) with the new unified update format, you can explicitly pass the pre or post-value to the custom SQL statement using the @BEFORE, @AFTER, and @BEFOREAFTER functions. Though Oracle GoldenGate 12.2.x attempts to use this new update format by default, the old format cam be preserved if there are conflicting parameters that would have previously generated two separate pre and post change records. In these cases, an informational message is logged in the report file.

Use the GETUPDATEBEFORES parameter as follows:

- in the Extract parameter file to extract before images from the data source.
- in the Replicat parameter file to include before images in a Replicat operation.

You can compare before images with after images to identify the net results of a transaction or perform other delta calculations. For example, if a BALANCE field is $100 before an update and $120 afterward, a comparison would show the difference of $20. You can use the column-conversion functions of Oracle GoldenGate to perform the comparisons and calculations.

To reference before images in the parameter file, use the @BEFORE conversion function. For example:

```
COLMAP (previous = @BEFORE (balance))
```

GETUPDATEBEFORES is required when using the Conflict Detection and Resolution (CDR) feature for a Oracle database on any of the platforms that are supported by Oracle GoldenGate. See *Administering Oracle GoldenGate* for more information about CDR.

The GETUPDATEBEFORES and IGNOREUPDATEBEFORES parameters are table-specific. One parameter remains in effect for all subsequent TABLE or MAP statements, until the other parameter is encountered.

Because you can selectively enable or disable these parameters between MAP statements, you can enable or disable them for different threads of a coordinated Replicat. Specify the GETUPDATEBEFORES threads in one set of MAP statements, and specify the IGNOREUPDATEBEFORES threads in a different set of MAP statements.

Default

```
IGNOREUPDATEBEFORES
```

Syntax

```
GETUPDATEBEFORES | IGNOREUPDATEBEFORES
```

Example

This example shows how you can apply GETUPDATEBEFORES and IGNOREUPDATEBEFORES selectively to different MAP statements, each of which represents a different thread of a coordinated Replicat.

```
GETUPDATEBEFORES
MAP sales.cust, TARGET sales.cust, THREAD (1);
MAP sales.ord, TARGET sales.ord, THREAD (2);
IGNOREUPDATEBEFORES
MAP sales.loc, TARGET sales.loc, THREAD (3);
```

2.75 GETUPDATES | IGNOREUPDATES

Valid For

Extract and Replicat

Description

Use the GETUPDATES and IGNOREUPDATES parameters to control whether or not Oracle GoldenGate processes UPDATE operations. These parameters are table-specific. One parameter remains in effect for all subsequent TABLE or MAP statements, until the other parameter is encountered.

Because you can selectively enable or disable these parameters between MAP statements, you can enable or disable them for different threads of a coordinated Replicat. Specify the GETUPDATES threads in one set of MAP statements, and specify the IGNOREUPDATES threads in a different set of MAP statements.

Default

```
GETUPDATES
```

Syntax

```
GETUPDATES | IGNOREUPDATES
```

Example

This example shows how you can apply GETUPDATES and IGNOREUPDATES selectively to different MAP statements, each of which represents a different thread of a coordinated Replicat.

```
GETUPDATES
MAP sales.cust, TARGET sales.cust, THREAD (1);
MAP sales.ord, TARGET sales.ord, THREAD (2);
IGNOREUPDATES
MAP sales.loc, TARGET sales.loc, THREAD (3);
```

2.76 GGSCHEMA

Valid For

GLOBALS

Description

Use the GGSCHEMA parameter to specify the name of the schema that contains the database objects that are owned by Oracle GoldenGate, such as those that support DDL replication for trigger-based replication, those that are a part of the heartbeat table implementation, and those that are part of the SQL Server CDC Capture and Cleanup implementation.. The schema name mentioned under GGSCHEMA is treated as system object, and Table names with wildcards under the GGSCHEMA are excluded from the Extract. If you need to capture and replicate objects in GGSCHEMA, do not use wildcards and make sure that you explicitly map the respective table names.

See Establishing Oracle GoldenGate Credentials in *Using Oracle GoldenGate for Oracle Database*for more information about the Oracle GoldenGate database objects.

Default

None

Syntax

```
GGSCHEMA [container.]schema_name
```

[container.]schema_name
The fully qualified name of the Oracle GoldenGate objects' schema. Use the full two-part name if the schema is within an Oracle container database.

Example

```
GGSCHEMA ogg
```

2.77 GROUPTRANSOPS

Valid For

Replicat

Description

Use the GROUPTRANSOPS parameter to control the number of SQL operations that are contained in a Replicat transaction when operating in its normal mode (non-BATCHSQL). Increasing the number of operations in a Replicat transaction improves the performance of Oracle GoldenGate by:

- Reducing the number of transactions executed by Replicat.
- Reducing I/O activity to the checkpoint file and the checkpoint table, if used. Replicat issues a checkpoint whenever it applies a transaction to the target, in addition to its scheduled checkpoints.

Replicat accumulates operations from source transactions, in transaction order, and applies them as a group within one transaction on the target. GROUPTRANSOPS sets a minimum value rather than an absolute value, to avoid splitting apart source transactions. Replicat waits until it receives all operations from the last source transaction in the group before applying the target transaction.

For example, if transaction 1 contains 200 operations, and transaction 2 contains 400 operations, and transaction 3 contains 500 operations, the Replicat transaction contains all 1,100 operations even though GROUPTRANSOPS is set to the default of 1,000. Conversely, Replicat might apply a transaction before reaching the value set by GROUPTRANSOPS if there is no more data in the trail to process.

Table 2-13 Replicat GROUPTRANSOPS

Source Transactions (assumes same table and column list)	Replicat transaction in normal (GROUPTRANSOPS) mode
Transaction 1:	**Transaction:**
INSERT	INSERT
DELETE	DELETE
Transaction 2:	INSERT
INSERT	DELETE
DELETE	INSERT
Transaction 3:	DELETE
INSERT	
DELETE	

Avoid setting GROUPTRANSOPS to an arbitrarily high number because the difference between source and target transaction boundaries can increase the latency of the target data.

(Oracle only) For an integrated Replicat, GROUPTRANSOPS is effective only when the integrated Replicat parameter PARALLELISM is set to 1.

Default

Nonintegrated Replicat: 1000 operations, Integrated Replicat: 50 operations

Syntax

```
GROUPTRANSOPS number
```

number
The minimum number of operations to be applied in a Replicat transaction. A value of 1 executes the operations within the same transaction boundaries as the source transaction. The value must be at least 1.

Example

```
GROUPTRANSOPS 2000
```

2.78 HANDLECOLLISIONS | NOHANDLECOLLISIONS

Valid For

Replicat

Description

Use the HANDLECOLLISIONS and NOHANDLECOLLISIONS parameters to control whether or not Replicat tries to resolve duplicate-record and missing-record errors when applying SQL on the target. These errors, called *collisions*, occur during an initial load, when data from source tables is being loaded to target tables while Oracle GoldenGate is replicating transactional changes that are being made to those tables. When Oracle GoldenGate applies the replicated changes after the load is finished, HANDLECOLLISIONS provides Replicat with error-handling logic for these collisions.

You can use HANDLECOLLISIONS and NOHANDLECOLLISIONS in the following ways:

- You can enable HANDLECOLLISIONS and NOHANDLECOLLISIONS in a global manner by specifying them at the root level of the parameter file. One parameter remains enabled for all subsequent MAP statements in the parameter file, until the opposing parameter is encountered.
- You can enable HANDLECOLLISIONS or NOHANDLECOLLISIONS within a specific MAP parameter to enable or disable error handling only for that source-target mapping.

The preceding methods can be combined. You can specify a global collisions-handling rule and then override that rule with different collisions-handling rules in the MAP statements. A MAP specification always overrides the global specification.

> **Note:**
>
> Error Handling of Integrated Replicat is not appropriate with HANDLECOLLISIONS. Oracle recommends that you use precise instantiation methods instead of using HANDLECOLLISIONS.

How HANDLECOLLISIONS Works

The following example explains how HANDLECOLLISIONS works:

- When Replicat encounters an update to a column that Oracle GoldenGate is using as a key, the handling is as follows:
  - If the row with the old key is not found in the target, the change record in the trail is converted to an insert.
  - If a row with the new key exists in the target, Replicat deletes the row that has the old key (it would not exist if the update had executed successfully), and then the row with the new key is updated as an overlay where the trail values replace the current values.

  This logic requires all of the columns in the table (not just the ones that changed) to be logged to the transaction log, either by default or by force, such as by using the COLS option of ADD TRANDATA for an Oracle database. See Possible Solutions to Avoid Missing Column Values.
- When Replicat encounters a duplicate-record error, the static record that was applied by the initial load is overwritten by the change record in the trail. Overlaying the change is safer from an operational standpoint than ignoring the duplicate-record error.
- Replicat with HANDLECOLLISIONS doesn't discard the change record in the trail even if update or delete operation doesn't affect a key column in the source and Replicat encounters a missing-record error in the target. These errors happen when a record is changed on the source system and then the record is deleted before the table data is extracted by the initial-load process. For example:
  1. The application updates record A in source table1.
  2. Extract extracts the update.
  3. The application deletes record A in source table1.
  4. Extract extracts the delete.

5. Oracle GoldenGate extracts initial-load data from source table1, without record A.
6. Oracle GoldenGate applies the initial load, without record A.
7. Replicat attempts to apply the update of record A.
8. The database returns a "record missing" error.
9. Replicat attempts to apply the delete of record A.
10. The database returns a "record missing" error.

Disable `HANDLECOLLLIONS` after the transactional changes captured during the initial load are applied to the target tables, so that Replicat does not automatically handle subsequent errors. Errors generated after initial synchronization indicate an abnormal condition and should be evaluated by someone who can determine how to resolve them. For example, a missing-record error could indicate that a record which exists on the source system was inadvertently deleted from the target system.

You can turn off `HANDLECOLLISIONS` in the following ways:

- Stop Replicat and remove `HANDLECOLLISIONS` from the Replicat parameter file (can cause target latency). Alternatively, you can edit the parameter file to add `NOHANDLECOLLISIONS` before the `MAP` statements for which you want to disable the error handling.
- While Replicat is running, run GGSCI and then use the `SEND REPLICAT` command with the `NOHANDLECOLLISIONS` option for the tables that you want to affect.

> **Note:**
>
> If using `SEND REPLICAT`, make certain to remove `HANDLECOLLISIONS` from the parameter file or add a `NOHANDLECOLLISIONS` parameter before starting another Replicat run, so that `HANDLECOLLISIONS` does not activate again.

Possible Solutions to Avoid Missing Column Values

When a database does not log all of the column values of a source table by default, there can be errors if the target table has `NOT NULL` constraints when Replicat attempts to convert a primary-key update to an insert. You can work around this scenario in the following ways:

- Use the `NOCOMPRESSUPDATES` parameter in the Extract parameter file to send all of the columns of the table to the trail, and configure the database to log all column values. By default, Extract only writes the primary key and the columns that changed to the trail. This is the safest method, because it writes the current values at the time when the operation is performed and eliminates the need for fetching.
- Use the `FETCHOPTIONS` parameter with the `FETCHPKUPDATECOLS` option in the Extract parameter file. This configuration causes Extract to fetch unavailable columns when a key column is updated on the source. A fetch is the *current* value, not necessarily the value at the time of a particular update, so there can be data integrity issues. See "FETCHOPTIONS" for more information and additional fetch options to handle unsuccessful fetches.

Getting More Information about Initial Loads

See *Administering Oracle GoldenGate* for more information about Oracle GoldenGate initial load methods.

Default

`NOHANDLECOLLISIONS`

Syntax

```
HANDLECOLLISIONS | NOHANDLECOLLISIONS [_ALLOWPKMISSINGROWCOLLISIONS]
[THREADS (threadID[, threadID][, ...][, thread_range[, thread_range][, ...])]
```

`HANDLECOLLISIONS`
Enables collision handling.

`_ALLOWPKMISSINGROWCOLLISIONS`
Use `HANDLECOLLISIONS` with `_ALLOWPKMISSINGROWCOLLISIONS` to skip primary-key `UPDATE` operations if the corresponding target row does not exist.

> **Note:**
>
> Skipping operations can cause data corruption. See the Description in this topic.

`NOHANDLECOLLISIONS`
Turns off collision handling.

`THREADS (threadID[, threadID][, ...][, thread_range[, thread_range][, ...])`
Enables `HANDLECOLLISIONS` for the specified threads. When used in a global `HANDLECOLLISIONS` statement at the root level of the parameter file, `HANDLECOLLISIONS` is enabled for the specified threads wherever they are in all `MAP` statements where . When used in a `HANDLECOLLISIONS` clause of a `MAP` statement, `HANDLECOLLISIONS` is enabled only for that `MAP` statement.

`threadID[, threadID][, ...]`
Specifies a thread ID or a comma-delimited list of threads in the format of `threadID, threadID, threadID`.

`thread_range[, thread_range][, ...]`
Specifies a range of threads in the form of `threadIDlow-threadIDhigh` or a comma-delimted list of ranges in the format of `threadIDlow-threadIDhigh, threadIDlow-threadIDhigh`.

A combination of these formats is permitted, such as `threadID, threadID, threadIDlow-threadIDhigh`.

Examples

Example 1
This example enables `HANDLECOLLISIONS` for all `MAP` statements in the parameter file.

```
HANDLECOLLISIONS
MAP hr.emp, TARGET hr.emp;
MAP hr.job_hist, TARGET hr.job_hist;
MAP hr.dep, TARGET hr.dep;
MAP hr.country, TARGET hr.country;
```

Example 2
This example enables HANDLECOLLISIONS for some MAP statements while disabling it for others.

```
HANDLECOLLISIONS
MAP hr.emp, TARGET hr.emp;
MAP hr.job_hist, TARGET hr.job_hist;
NOHANDLECOLLISIONS
MAP hr.dep, TARGET hr.dep;
MAP hr.country, TARGET hr.country;
```

Example 3
This example shows the basic use of HANDLECOLLISIONS within a MAP statement.

```
MAP dbo.tcust, TARGET dbo.tcust, HANDLECOLLISIONS;
```

Example 4
This example shows a combination of global and MAP-level use. The MAP specification overrides the global specification for the specified tables.

```
HANDLECOLLISIONS
MAP hr.emp, TARGET hr.emp;
MAP hr.job_hist, TARGET hr.job_hist;
MAP hr.dep, TARGET hr.dep, NOHANDLECOLLISIONS;
MAP hr.country, TARGET hr.country, NOHANDLECOLLISIONS;
```

Example 5
In the following example, HANDLECOLLISIONS is enabled globally for all MAP statements, except for default thread 0 in the first MAP statement and for thread 3 in the second MAP statement.

```
HANDLECOLLISIONS
MAP fin.*, TARGET fin.*;
MAP sales.*, TARGET sales.*;
MAP orders.*, TARGET orders.*;
MAP scott.cust, TARGET scott.cust, NOHANDLECOLLISIONS;
MAP amy.cust, TARGET amy.cust, THREAD(3), NOHANDLECOLLISIONS;
```

Example 6
In this example, HANDLECOLLISIONS is enabled globally, but turned off for thread 3. The remaining threads 1, 2, and 4 will handle collisions.

```
HANDLECOLLISIONS
NOHANDLECOLLISIONS THREAD(3)
MAP scott.emplyees, TARGET scott.employees, THREADRANGE(1,4, OID);
MAP scott.inventory, TARGET scott.inventory, THREADRANGE(1,4, OID);
MAP scott.cust, TARGET scott.cust, THREADRANGE(1,4, OID);
```

Example 7
In this example, HANDLECOLLISIONS is enabled globally, then disabled globally for threads 5 through 7. In the first map statement, all threads will handle collisions, since the HANDLECOLLISIONS parameter does not specify a thread or a range. In the second map statement, only threads 4, 8, and 9 will handle collisions, because the global NOHANDLECOLLISIONS applies to threads 5-7.

```
HANDLECOLLISIONS
NOHANDLECOLLISIONS THREADRANGE(5-7)
MAP scott.cust, TARGET scott.cust, THREADRANGE(4,9,OID), HANDLECOLLISIONS;
MAP scott.offices, TARGET scott.offices, THREADRANGE(4,9,OID);
MAP scott.emp, TARGET scott.emp, THREADRANGE(4,9,OID);
MAP scott.ord, TARGET scott.ord, THREADRANGE(4,9,OID);
MAP acct.*, TARGET acct.*;
MAP admin.*, TARGET admin.*;
```

2.79 HANDLETPKUPDATE

Valid For

Replicat (nonintegrated mode)

Description

Use the HANDLETPKUPDATE parameter to prevent constraint errors when an update to a primary key results in a transient duplicate. This is an Oracle parameter for nonintegrated Replicat and is required if the target database is any version earlier than Oracle version 11.2.0.2. For target Oracle databases that are version 11.2.0.2 or later, transient primary-key duplicates are handled automatically without requiring HANDLETPKUPDATE. Integrated Replicat handles this issue automatically so it is not necessary to set this parameter

A transient primary-key duplicate occurs when an update affects the primary keys of multiple rows in a transaction. This kind of statement typically uses a SET x = x+*n* formula or other manipulation that shifts the values so that a new value is the same as an existing one.

The following example illustrates a sequence of value changes that can cause this condition. The example assumes table ITEM where the primary key column is named CODE and the current key values for the rows in the table are 1, 2, and 3.

```
update item set code = 2 where code = 1;
update item set code = 3 where code = 2;
update item set code = 4 where code = 3;
```

In this example, when the first UPDATE is applied to the target, there is an error because the primary key value of 2 already exists in the target. The target transaction returns constraint violation errors. By default, Replicat does not detect or handle these violations and abends.

When using HANDLETPKUPDATE, create the constraints as DEFERRABLE INITIALLY IMMEDIATE on the target tables. If the target constraints cannot be DEFERRABLE, Replicat handles the errors according to existing rules specified with the HANDLECOLLISIONS and REPERROR parameters, or else it abends.

This parameter can be used in a parameter file, and it can be used within a MAP statement as follows:

```
MAP ggs.equip_account, TARGET ggs.equip_account, HANDLETPKUPDATE;
```

Default

Abend on transient primary key updates

Syntax

```
HANDLETPKUPDATE
```

2.80 HAVEUDTWITHNCHAR

Valid For

Replicat (Oracle only)

Description

Use the HAVEUDTWITHNCHAR parameter when the source data contains user-defined types that have an NCHAR, NVARCHAR2, or NCLOB attribute. When this data is encountered in the trail, HAVEUDTWITHNCHAR causes Replicat to connect to the Oracle target in AL32UTF8, which is required when a user-defined data type contains one of those attributes.

HAVEUDTWITHNCHAR is not required if the character set of the target is AL32UTF8. However, it is required if only NLS_LANG is set to AL32UTF8 on the target. By default Replicat ignores NLS_LANG and connects to an Oracle database in the native character set of the database. Replicat uses the OCIString object of the Oracle Call Interface, which does not support NCHAR, NVARCHAR2, or NCLOB attributes, so Replicat must bind them as CHAR. Connecting to the target in AL32UTF8 prevents data loss in this situation.

HAVEUDTWITHNCHAR must be specified before the USERID or USERIDALIAS parameter in the parameter file.

Default

None

Syntax

```
HAVEUDTWITHNCHAR
```

2.81 HEARTBEATTABLE

Valid For

GLOBALS

Description

Use HEARTBEATTABLE to specify a non-default name of the heartbeat table.
The table name GG_HEARTBEAT is the default. This name used to denote the heartbeat table is used to create a seed and history table, GG_HEARTBEAT_SEED and GG_HEARTBEAT_HISTORY respectively. Specifying one name reserves all names used by the heartbeat infrastructure. If the schema name is not specified, the value in GGSCHEMA is used for schema name.

Default

None

Syntax

```
HEARTBEATTABLE schema_name.heartbeat_table_name
```

schema_name
The name of the schema you want to use with the heartbeat table. This is not needed if you have specified the schema using the GGSCHEMA parameter in your GLOBALS file.

heartbeat_table_name
The heartbeat table name you want to use. The default table name is GG_HEARTBEAT.

2.82 INCLUDE

Valid For

Extract and Replicat

Description

Use the INCLUDE parameter to include a macro library in a parameter file. See *Administering Oracle GoldenGate* for more information about using macros.

Default

None

Syntax

```
INCLUDE library
```

library
The relative or full path to library file.

Example

The following example includes macro library mdatelib.mac.

```
INCLUDE /ggs/dirprm/mdatelib.mac
```

2.83 INSERTALLRECORDS

Valid For

Replicat

Description

Use the INSERTALLRECORDS parameter to keep a record of all operations made to a target record, instead of maintaining just the current version. INSERTALLRECORDS causes Replicat to insert every change that is made to a record as a new record in the database. The initial insert and subsequent updates and deletes are maintained as point-in-time snapshots.

Some cases for using INSERTALLRECORDS are the following:

- To work within an exceptions MAP statement. In an exceptions MAP statement, INSERTALLRECORDS causes the values of operations that generated errors to be inserted as new records in an exceptions table as part of an error-handling strategy.
- To maintain a transaction history. By inserting every change to a specific row as a new record in the database, you can maintain a history of all changes made to that row, instead of maintaining just the current version. Each insert is a point-in-time snapshot that can be queried as needed for auditing purposes. Combining historical data with special transaction information provides a way to create a more useful target reporting database.

INSERTALLRECORDS can be used at the root level of the parameter file to affect all subsequent MAP statements, and it can be used within a MAP statement to affect a specific table or multiple tables specified with a wildcard.

Getting More Information about INSERTALLRECORDS

See *Administering Oracle GoldenGate* for information about creating a transaction history table.

See *Administering Oracle GoldenGate* for information about using an exceptions MAP statement.

See "TABLE | MAP" for MAP syntax.

Default

None

Syntax

```
INSERTALLRECORDS
```

Examples

Example 1
This example shows INSERTALLRECORDS at the root level of the parameter file as part of an exceptions handling configuration.

```
REPLICAT deliv
USERIDALIAS tiger1
ASSUMETARGETDEFS
REPERROR (DEFAULT, EXCEPTION)
MAP ggs.equip_account, TARGET ggs.equip_account2,
COLMAP (USEDEFAULTS);
MAP ggs.equip_account, TARGET ggs.equip_account_exception,
EXCEPTIONSONLY,
INSERTALLRECORDS
COLMAP (USEDEFAULTS,
DML_DATE = @DATENOW(),
OPTYPE = @GETENV('LASTERR', 'OPTYPE'),
DBERRNUM = @GETENV('LASTERR', 'DBERRNUM'),
DBERRMSG = @GETENV('LASTERR', 'DBERRMSG'));
```

Example 2
This example shows INSERTALLRECORDS in a MAP statement.

```
REPLICAT deliv
USERIDALIAS tiger1
SOURCEDEFS /ggs/dirdef/defs
REPERROR DEFAULT, ABEND
MAP fin.accTAB, TARGET fin.custTAB, INSERTALLRECORDS;
```

2.84 INSERTAPPEND | NOINSERTAPPEND

Valid For

Replicat (Oracle Nonintegrated mode)

Description

Use the INSERTAPPEND and NOINSERTAPPEND parameters to control whether or not a Replicat operating in nonintegrated mode uses an APPEND hint when it applies INSERT operations (used for array binding) to Oracle target tables. These parameters are valid only for Oracle databases and are only compatible with BATCHSQL mode.

INSERTAPPEND causes Replicat to use the APPEND_VALUES hint when it applies INSERT operations to Oracle target tables. It is appropriate for use as a performance improvement when the replicated transactions are large and contain multiple inserts into the same table. If the transactions are small, using INSERTAPPEND can cause a performance decrease. For more information about when APPEND hints should be used, consult the Oracle documentation.

The BATCHSQL parameter must be used when using INSERTAPPEND. Replicat will abend if BATCHSQL is not used.

These parameters can be used in two ways: When used as standalone parameters at the root of the parameter file, one remains in effect for all subsequent TABLE or MAP statements, until the other is encountered. When used within a MAP statement, they override any standalone INSERTAPPEND or NOINSERTAPPEND entry that precedes the MAP statement.

See "TABLE | MAP" for more information about the MAP parameter.

Default

```
NOINSERTAPPEND
```

Syntax

```
INSERTAPPEND | NOINSERTAPPEND
```

Examples

Example 1
The following is part of a Replicat parameter file that shows how INSERTAPPEND is used for all of the tables in the fin schema, except for the inventory table.

```
BATCHSQL
INSERTAPPEND
MAP fin.*, TARGET fin2.*;
MAPEXCLUDE fin.inventory;
NOINSERTAPPEND
MAP fin.inventory, TARGET fin2.inventory;
```

Example 2
The following is part of a Replicat parameter file that shows how INSERTAPPEND is used for all of the tables in the MAP statements, except for the inventory table.

```
BATCHSQL
MAP fin.orders, TARGET fin.orders;
MAP fin.customers, TARGET fin.customers;
MAP fin.inventory, TARGET fin.inventory, NOINSERTAPPEND;
```

2.85 INSERTDELETES | NOINSERTDELETES

Valid For

Replicat

Description

Use the INSERTDELETES and NOINSERTDELETES parameters to control whether or not Oracle GoldenGate converts source delete operations to insert operations on the target database. The parameters are table-specific. One parameter remains in effect for all subsequent MAP statements, until the other parameter is encountered.

When using INSERTDELETES, use the NOCOMPRESSDELETES parameter so that Extract does not compress deletes.

Because you can selectively enable or disable these parameters between MAP statements, you can enable or disable them for different threads of a coordinated Replicat. Specify the INSERTDELETES threads in one set of MAP statements, and specify the NOINSERTDELETES threads in a different set of MAP statements.

Default

```
NOINSERTDELETES
```

Syntax

```
INSERTDELETES | NOINSERTDELETES
```

Example

This example shows how you can apply INSERTDELETES and NOINSERTDELETES selectively to different MAP statements, each of which represents a different thread of a coordinated Replicat.

```
INSERTDELETES
MAP sales.cust, TARGET sales.cust, THREAD (1);
MAP sales.ord, TARGET sales.ord, THREAD (2);
NOINSERTDELETES
MAP sales.loc, TARGET sales.loc, THREAD (3);
```

2.86 INSERTMISSINGUPDATES | NOINSERTMISSINGUPDATES

Valid For

Replicat

Description

Use the INSERTMISSINGUPDATES and NOINSERTMISSINGUPDATES parameters to control whether or not Oracle GoldenGate inserts a record based on the source record when the target record does not exist.

INSERTMISSINGUPDATES inserts the missing update but should only be used when the source database logs all column values, whether or not they changed). It can work with a database that uses a compressed form of updates (where only the changed values are logged) if the target database allows NULL to be used for the missing column values.

When the default of NOINSERTMISSINGUPDATES is in effect, a missing record causes an error, and the transaction may abend depending on REPERROR settings.

The INSERTMISSINGUPDATES and NOINSERTMISSINGUPDATES parameters are table-specific. One parameter remains in effect for all subsequent MAP statements, until the other parameter is encountered.

Default

```
NOINSERTMISSINGUPDATES
```

Syntax

```
INSERTMISSINGUPDATES | NOINSERTMISSINGUPDATES
```

2.87 INSERTUPDATES | NOINSERTUPDATES

Valid For

Replicat

Description

Use the INSERTUPDATES and NOINSERTUPDATES parameters to control whether or not Oracle GoldenGate converts update operations to insert operations. For updates to be converted to inserts, the database must log all column values either by default or by means of supplemental logging.

The parameters are table-specific. One parameter remains in effect for all subsequent MAP statements, until the other parameter is encountered.

To ensure that updates are not compressed by Extract when using INSERTUPDATES, use the NOCOMPRESSUPDATES parameter.

Because you can selectively enable or disable these parameters between MAP statements, you can enable or disable them for different threads of a coordinated Replicat. Specify the INSERTUPDATES threads in one set of MAP statements, and specify the NOINSERTUPDATES threads in a different set of MAP statements.

Default

```
NOINSERTUPDATES
```

Syntax

```
INSERTUPDATES | NOINSERTUPDATES
```

Example

This example shows how you can apply INSERTUPDATES and NOINSERTUPDATES selectively to different MAP statements, each of which represents a different thread of a coordinated Replicat.

```
INSERTUPDATES
MAP sales.cust, TARGET sales.cust, THREAD (1);
MAP sales.ord, TARGET sales.ord, THREAD (2);
NOINSERTUPDATES
MAP sales.loc, TARGET sales.loc, THREAD (3);
```

2.88 LIST | NOLIST

Valid For

Extract and Replicat

Description

Use the LIST and NOLIST parameters 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 and NOLIST parameters within the parameter file or within the macro library file. Using NOLIST reduces the size of the report file. For more information about using macros, see the *Administering Oracle GoldenGate*.

Default

```
LIST
```

Syntax

```
LIST | NOLIST
```

Example

In the following example, NOLIST excludes the macros in the hugelib macro library from being listed in the report. Using LIST after the INCLUDE statement restores normal listing for subsequent macros.

```
NOLIST
INCLUDE /ggs/hugelib.mac
LIST
```

2.89 LOBMEMORY

Valid For

Extract and Replicat for DB2 z/OS

Description

Use the LOBMEMORY parameter to control the amount of memory and temporary disk space available for caching transactions that contain LOBs. Because Oracle GoldenGate applies only committed transactions to the target database, it requires sufficient system memory to store LOB data until either a commit or rollback indicator is received.

This parameter is for use with a DB2 z/OS database. For all other databases, use the CACHEMGR parameter.

About Memory Management with LOBMEMORY

LOBMEMORY enables you to tune the cache size of Oracle GoldenGate for LOB transactions and define a temporary location on disk for storing data that exceeds the size of the cache. Options are available for defining the total cache size, the per-transaction memory size, the initial and incremental memory allocation, and disk storage space.

LOB transactions are added to the memory pool specified by RAM, and each is flushed to disk when TRANSRAM is reached. An initial amount of memory is allocated to each transaction based on INITTRANSRAM and is increased by the amount specified by RAMINCREMENT as needed, up to the maximum set with TRANSRAM. Consequently, the value for TRANSRAM should be evenly divisible by the sum of (INITTRANSRAM + RAMINCREMENT).

Default

See option defaults

Syntax

```
LOBMEMORY
[RAM size]
[TRANSRAM size]
[TRANSALLSOURCES size]
[INITTRANSRAM size]
[RAMINCREMENT size]
[DIRECTORY (directory, max_directory_size, max_file_size)]
```

RAM *size*
Specifies the total amount of memory to use for all cached LOB transactions. The default is 200 megabytes. The value can be specified in bytes or in terms of gigabytes, megabytes, or kilobytes in any of the following forms:
GB | MB | KB | G | M | K | gb | mb | kb | g | m | k

TRANSRAM *size*
Specifies the total amount of memory to use for a single LOB transaction. The default is 50 megabytes. The value can be specified in bytes or in terms of gigabytes, megabytes, or kilobytes in any of the following forms:
GB | MB | KB | G | M | K | gb | mb | kb | g | m | k
TRANSRAM should be evenly divisible by both INITTRANSRAM and RAMINCREMENT for optimal results.

TRANSALLSOURCES *size*
Specifies the total amount of memory and disk space to use for a single LOB transaction. The default is 50% of total available memory (memory and disk). The value can be specified in bytes or in terms of gigabytes, megabytes, or kilobytes in any of the following forms:
GB | MB | KB | G | M | K | gb | mb | kb | g | m | k

INITTRANSRAM *size*
Specifies the initial amount of memory to allocate for a LOB transaction. The default is 500 kilobytes. The value can be specified in bytes or in terms of gigabytes, megabytes, or kilobytes in any of the following forms:
GB | MB | KB | G | M | K | gb | mb | kb | g | m | k

RAMINCREMENT *size*
Specifies the amount of memory to increment when a LOB transaction requires more memory. The default is 500 kilobytes. The value can be specified in bytes or in terms of gigabytes, megabytes, or kilobytes in any of the following forms:
GB | MB | KB | G | M | K | gb | mb | kb | g | m |

DIRECTORY (*directory, max_directory_size, max_file_size*)
Specifies temporary disk storage for LOB transaction data when its size exceeds the maximum specified with TRANSRAM. You can specify DIRECTORY more than once.

- *directory* is the fully qualified name of a directory. The default is the dirtmp sub-directory of the Oracle GoldenGate directory.
- *max_directory_size* is the maximum size of all files in the directory. The default is 2 gigabytes. If the space specified is not available, then 75% of available disk space is used.
- *max_file_size* is the maximum size of each file in the directory. The default is 200 megabytes.

Values can be specified in bytes or in terms of gigabytes, megabytes, or kilobytes in any of the following forms:
GB | MB | KB | G | M | K | gb | mb | kb | g | m | k
The directory size and file size must be greater than the size of the memory specified with RAM.
The file names use the following format.
*group*_blob_00001.mem
or...
*PID*_blob_00001.mem
A group name is used for online processes. A system process ID number (PID) is used for one-time runs specified with the SPECIALRUN parameter.
The format for a threaded Extract is similar to the following, depending on the database.
group_thread #_00001.mem

Examples

Example 1
The following example allows per-transaction memory to be incremented ten times before data is flushed to disk, once for the initial allocation specified with INITTRANSRAM and then nine more times as permitted by RAMINCREMENT.

```
LOBMEMORY DIRECTORY (c:\test\dirtmp, 3000000000, 300000000), &
RAM 8000K, TRANSRAM 1000K, INITTRANSRAM 100K, RAMINCREMENT 100K
```

Example 2
The following is the same as the preceding example, but with the addition of a second directory.

```
LOBMEMORY DIRECTORY (c:\test\dirtmp, 3000000000, 300000000), &
DIRECTORY (c:\test\dirtmp2, 1000000000, 5000000), &
RAM 8000K, TRANSRAM 1000K, INITTRANSRAM 100K, RAMINCREMENT 100K
```

2.90 LOGALLSUPCOLS

Valid For

Extract

Description

Use the LOGALLSUPCOLS parameter to control the writing of supplementally logged columns specified with ADD TRANDATA or ADD SCHEMATRANDATA to the trail.

LOGALLSUPCOLS supports integrated Replicat (for Oracle database) and the Oracle GoldenGateConflict Detection and Resolution feature (CDR). The supplementally logged columns are a union of the scheduling columns that are required to ensure data integrity across parallel Replicat threads and the conflict detection and resolution (CDR) columns. Scheduling columns are primary key, unique index, and foreign key columns. Including all of these supplementally logged columns satisfies the requirements of both CDR and dependency computation in parallel Replicat processing.

LOGALLSUPCOLS causes Extract to do the following with these supplementally logged columns:

- Automatically includes in the trail record the before image for UPDATE operations.
- Automatically includes in the trail record the before image of all supplementally logged columns for both UPDATE and DELETE operations.

For Extract versions older than 12c, you can use GETUPDATEBEFORES and NOCOMPRESSDELETES parameters to satisfy the same requirement. See GETUPDATEBEFORES | IGNOREUPDATEBEFORES and COMPRESSUPDATES | NOCOMPRESSUPDATES for more information.

LOGALLSUPCOLS | NOLOGALLSUPCOLS takes precedence over the following parameters, if used:

- GETUPDATEBEFORES | IGNOREUPDATEBEFORES
- COMPRESSDELETES | NOCOMPRESSDELETES
- COMPRESSUPDATES | NOCOMPRESSUPDATES for before images, but COMPRESSUPDATES | NOCOMPRESSUPDATES takes precedence over LOGALLSUPCOLS on after images.

Default

LOGALLSUPCOLS

Syntax

LOGALLSUPCOLS

2.91 LOOK_AHEAD_TRANSACTIONS

Valid For

Parallel Replicat

Description

This is an advanced parameter. It controls how far ahead the Scheduler looks when batching transactions.

Default

The default value is 10000.

Syntax

```
LOOK_AHEAD_TRANSACTIONS
```

2.92 MACRO

Valid For

Extract and Replicat

Description

Use the `MACRO` parameter to create an Oracle GoldenGate macro. See *Administering Oracle GoldenGate* for more information about using macros, including how to invoke them properly.

Default

None

Syntax

The following must be used in the order shown:

```
MACRO #macro_name
PARAMS (#param_name [, ...])
BEGIN
macro_body
END;
```

`MACRO`
Starts the macro specification.

`#`
The macro character. Macro and parameter names must begin with a macro character. Anything in the parameter file that begins with the macro character is assumed to be either a macro or a macro parameter.
The default macro character is the pound (#) character, as in the following examples:

```
MACRO #macro1
PARAMS (#param1, #param2)
```

You can change the macro character with the MACROCHAR parameter.

macro_name
The name of the macro. Macro names must be one word with alphanumeric characters (underscores are allowed) and are not case-sensitive. Each macro name in a parameter file must be unique. Do not use quotes, or else the macro name will be treated as text and ignored.

PARAMS
Starts a parameter clause. A parameters clause is optional. The maximum size and number of parameters is unlimited, assuming sufficient memory is available.

param_name
Describes a parameter to the macro. Parameter names are not case-sensitive. Do not use quotes, or else the parameter name will be treated as text and ignored.
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.

BEGIN
Begins the macro body. Must be specified before the macro body.

macro_body
The body of the macro. The size of the macro body is unlimited, assuming sufficient available memory. A macro body can include any of the following types of statements:

- Simple parameter statements, as in:

  ```
  COL1 = COL2
  ```

- Complex statements, as in:

  ```
  COL1 = #val2
  ```

- Invocations of other macros, as in:

  ```
  #colmap(COL1, #sourcecol)
  ```

END;
Concludes the macro definition. The semicolon is required to complete the definition.

Examples

Example 1
The following example defines a macro that takes parameters.

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

Example 2
The following example defines a macro that does not require parameters.

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

Example 3
The following example defines a macro named `#assign_date` that calls another macro named `#make_date`.

```
MACRO #assign_date
PARAMS (#target_col, #year, #month, #day)
BEGIN
#target_col = #make_date (#year, #month, #day)
END;
```

2.93 MACROCHAR

Valid For

Extract and Replicat

Description

Use the `MACROCHAR` parameter to change the macro character of a macro definition to something other than the # character. You might need to change the macro character when, for example, table names include the # character.

The `MACROCHAR` parameter can only be used once in the parameter file. Place the `MACROCHAR` parameter before the first `MACRO` parameter in the parameter file. Anything in the parameter file that begins with the specified macro character is assumed to be either a macro or a macro parameter. All macro definitions in the parameter file must use the specified character.

`MACROCHAR` cannot be used with query parameters.

See also "MACRO".

See the *Administering Oracle GoldenGate* for more information about using macros.

Default

(pound symbol)

Syntax

```
MACROCHAR character
```

`character`
The character to be used as the macro character. Valid user-defined macro characters are letters, numbers, and special characters such as the ampersand (`&`) or the underscore (_).

Example

In the following example, $ is defined as the macro character.

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

2.94 MAP for Extract

Valid For

Extract

Description

Use the MAP parameter for Extract when Extract is operating in classic capture mode and you need to use the ALTID component of this parameter to map an object ID to an object name. ALTID specifies the correct object ID if Extract is capturing from Oracle transaction logs that were generated by a database other than the one to which Extract is connected. This configuration is required when Extract is not permitted to connect directly to the production (source) database to capture production transactions.

When Extract cannot connect directly to a source database, it connects to a live standby or other facsimile database, but it reads transaction logs that are sent from the source database. By querying the catalog of the alternate database, Extract can get the metadata that it needs to expand the transaction data into valid SQL statements, but it cannot use the object ID from this query. The local object ID for a table is different from the object ID of that table in the source database (and, thus, in the transaction log). You must manually map each table name to the source object ID by using a MAP statement with ALTID.

To Use MAP with ALTID

- Create one MAP statement with ALTID for each table that you want to capture. Wildcarded table names are not allowed for a MAP parameter that contains ALTID.
- To specify other processing for the same table (or tables), such as data filtering or manipulation, you must also create a TABLE statement for each of those tables. Wildcarding can be used to specify multiple tables with one TABLE statement, if appropriate.
- Use a regular Replicat MAP statement in the Replicat parameter file, as usual. MAP for Extract does not substitute for MAP for Replicat, which is required to map source tables to target tables.
- DDL capture and replication is not supported when using ALTID.

Default

None

Syntax

MAP *[container.]schema.table*, ALTID *object_ID* [, *object_ID*]

[container.]schema.table
The fully qualified name of the source table.

object_ID
The object ID of the table as it exists in the production (source) database.
If a table is partitioned, you can list the object IDs of the partitions that you want to replicate, separating each with a comma.

Examples

Example 1
This example maps a non-partitioned table or just one partition of a partitioned table.

```
MAP QASOURCE.T2, ALTID 75740;
```

Example 2
This example maps partitions of a partitioned table.

```
MAP QASOURCE.T_P1, ALTID 75257,75258;
```

2.95 MAP

See "TABLE | MAP".

2.96 MAPALLCOLUMNS| NOMAPALLCOLUMNS

Valid For

Valid as a standalone Replicat parameter or as an option to `MAP`.

Description

An Extract or Replicat checks if all source columns are mapped directly to the target, without using the column mapping function when `MAPALLCOLUMNS` parameter is specified. If any source column is not mapped, then the Extract or Replicat abends.

`MAPALLCOLUMNS` and `NOMAPALLCOLUMNS` can be used in two different ways. When specified at a global level, one parameter remains in effect for all subsequent MAP statements, until the other parameter is specified. When used within a `MAP` statement, they override the global specifications.

Default

```
NOMAPALLCOLUMNS
```

Syntax

```
MAPALLCOLUMNS | NOMAPALLCOLUMNS
```

Examples

Example 1
This example enables `MAPINVISIBLECOLUMNS` for some `MAP` statements while disabling it for others.

```
MAPALLCOLUMNS
MAP hr.emp, TARGET hr.emp2;
NOMAPALLCOLUMNS
MAP hr.dep, TARGET hr.dep2;
```

Example 2
This example shows a combination of global and MAP-level use of MAPINVISIBLECOLUMNS. The MAP specification overrides the global specification for the specified table.

```
NOMAPALLCOLUMNS
MAP hr.dep, TARGET hr.dep2;
MAP hr.emp, TARGET hr.emp2, MAPALLCOLUMNS ;
```

Example 3 Abend with the option because source col2 is not mapped
Source has col1 and col2. Target has col1

Example 4 Abend with the option because col2 is not mapped
Source and target both have col1 and col2 MAP source, TARGET target, COLMAP(col1 = col1, col2 = 'abc')

Example 5 Abend with the option because col2 is not mapped to the directory
Source and target both have col1 and col2. MAP source, TARGET target, COLMAP(col1 = col1, col2 = @STRCAT(col2, ' '))

2.97 MAP_PARALLELISM

Valid for

Parallel Replicat

Description

Configures number of mappers. It controls the number of threads used to read the trail file. The minimum value is 1, maximum value is 100 and the default value is 2.

Syntax

```
MAP_PARALLELISM value
```

Examples

```
MAP_PARALLELISM 3
```

2.98 MAPEXCLUDE

Valid For

Replicat

Description

Use the MAPEXCLUDE parameter with the MAP parameter to explicitly exclude source tables and sequences from a wildcard specification. You can use multiple MAPEXCLUDE statements for specific MAP statements.

MAPEXCLUDE is evaluated before evaluating the associated MAP parameters. Thus, the order in which they appear does not make a difference.

When using wildcards, be careful not to place them such that all objects are excluded, leaving nothing to process. For example, the following example captures nothing from cat1:

```
MAP cat1.schema*.tab*, TARGET schema*.tab*;
MAPEXCLUDE cat1.*.*
```

See also the EXCLUDEWILDCARDOBJECTSONLY parameter.

The default for resolving wildcards is WILDCARDRESOLVE DYNAMIC. Therefore, if a table that is excluded with MAPEXCLUDE is renamed to a name that satisfies a wildcard, the data will be captured. The DYNAMIC setting enables new table names that satisfy a wildcard to be resolved as soon as they are encountered and included in the Oracle GoldenGate configuration immediately. For more information, see WILDCARDRESOLVE.

Default

None

Syntax

```
MAPEXCLUDE [container.]owner.{table | sequence}
```

***container*.**
If the source database requires three-part names, specifies the name or wildcard specification of the Oracle container that contains the object to exclude.

owner
Specifies the name or wildcard specification of the owner, such as the schema, of the object to exclude.

table* | *sequence
The name or wildcard specification of the source object to exclude. To specify object names and wildcards correctly, see *Administering Oracle GoldenGate*.

Example 1

In this example, the source tables from catalog pdb1 with schema test beginning with tab and the source table pdb2.fin.acct are excluded from the trail files:

```
MAPEXCLUDE pdb1.test.tab*
MAP pdb1.*.*, TARGET *.*;
MAPEXCLUDE pdb2.fin.acct
MAP pdb2.*.*, TARGET *.*;
```

Example 2

The following example excludes all source tables from catalog beginning with pdb, that is, it excludes all tables from pdb1, pdb2, pdb3 and so on:

```
MAP pdb1.*.*, TARGET *.*;
MAP pdb2.*.*, TARGET *.*;
MAPEXCLUDE pdb1.test.tab*
MAPEXCLUDE pdb*.*.*
MAPEXCLUDE pdb2.fin.acct
```

2.99 MAPINVISIBLECOLUMNS | NOMAPINVISIBLECOLUMNS

Valid For

Replicat on Oracle. Valid as a standalone parameter or as an option to `MAP`.

Description

Use `MAPINVISIBLECOLUMNS` and `NOMAPINVISIBLECOLUMNS` to control whether or not Replicat includes invisible columns in Oracle target tables for default column mapping. For invisible columns in Oracle target tables that use explicit column mapping, they are always mapped so do not require this option.

`MAPINVISIBLECOLUMNS` and `NOMAPINVISIBLECOLUMNS` can be used in two different ways. When specified at a global level, one parameter remains in effect for all subsequent MAP statements, until the other parameter is specified. When used within a `MAP` statement, they override the global specifications

Default

```
NOMAPINVISIBLECOLUMNS
```

Syntax

```
MAPINVISIBLECOLUMNS | NOMAPINVISIBLECOLUMNS
[, THREAD (threadID[, threadID][, ...][, thread_range[, thread_range][, ...])]
```

`THREADS (`*`threadID`*`[, `*`threadID`*`][, ...][, `*`thread_range`*`[, `*`thread_range`*`] [, ...])`
Specifies `MAPINVISIBLECOLUMNS | NOMAPINVISIBLECOLUMNS` only for the specified thread or threads of a coordinated Replicat.

***`threadID`*`[, `*`threadID`*`][, ...]`**
Specifies a thread ID or a comma-delimited list of threads in the format of `threadID, threadID, threadID`.

`[, `*`thread_range`*`[, `*`thread_range`*`][, ...]`
Specifies a range of threads in the form of `threadIDlow-threadIDhigh` or a comma-delimted list of ranges in the format of `threadIDlow-threadIDhigh, threadIDlow-threadIDhigh`.

A combination of these formats is permitted, such as threadID, threadID, `threadIDlow-threadIDhigh`.

Examples

Example 1
This example enables `MAPINVISIBLECOLUMNS` for some `MAP` statements while disabling it for others.

```
MAPINVISIBLECOLUMNS
MAP hr.emp, TARGET hr.emp2;
```

```
NOMAPINVISIBLECOLUMNS
MAP hr.dep, TARGET hr.dep2;
```

Example 2
This example shows a combination of global and MAP-level use of MAPINVISIBLECOLUMNS. The MAP specification overrides the global specification for the specified table.

```
NOMAPINVISIBLECOLUMNS
MAP hr.dep, TARGET hr.dep2;
MAP hr.emp, TARGET hr.emp2, MAPINVISIBLECOLUMNS;
```

Example 3
In this example, MAPINVISIBLECOLUMNS is enabled globally, but turned off for thread 3. The remaining threads 1, 2, and 4 will include invisible target columns in default column mapping.

```
MAPINVISIBLECOLUMNS
NOMAPINVISIBLECOLUMNS THREAD(3)
MAP hr.dep, TARGET hr.dep2, THREADRANGE(1, 4);
MAP hr.emp, TARGET hr.emp2, THREADRANGE(1, 4);
```

2.100 MARKERTABLE

Valid For

GLOBALS

Description

Use the MARKERTABLE parameter to specify the name of the DDL marker table, if other than the default of GGS_MARKER. The marker table stores information about DDL operations.

The name of the marker table must also be specified with the marker_table_name parameter in the params.sql script. This script resides in the root Oracle GoldenGate installation directory.

This parameter is only valid for an Oracle database in which the capture configuration uses the Oracle GoldenGate DDL trigger to support DDL replication. For more information about the Oracle GoldenGate DDL objects, see Installing Trigger-Based DDL Capture in *Using Oracle GoldenGate for Oracle Database*.

Default

```
GGS_MARKER
```

Syntax

```
MARKERTABLE [container.]table_name
```

[container.]table_name
The fully qualified three-part or two-part name of the marker table. To specify object names and wildcards correctly, see *Administering Oracle GoldenGate*.

2.101 MARKERTABLE

Valid For

GLOBALS

Description

`MASTERKEYNAME` controls the name of the masterkey that Oracle GoldenGate processes in a deployment will use to retrieve the key from the wallet. If no masterkey is provided, the default value is `OGG_DEFAULT_MASTERKEY`. The non-mandatory option `VERSION` takes one number between 1 and 65535 (0xffff). When present, it forces the Oracle GoldenGate processes in the deployment to use that particular version of the masterkey to encrypt or decrypt trails. This is not needed during normal operation, but might be useful when debugging old trail files if the key has been rolled over since the date the old trail was created.

Default

`OGG_DEFAULT_MASTERKEY`

Syntax

```
MASTERKEYNAME [VERSION]
```

2.102 MAXDISCARDRECS

Valid For

Extract and Replicat

Description

Use the `MAXDISCARDRECS` parameter to limit the number of errors that are reported to the discard file per `MAP` statement.

Use this parameter for the following reasons:

- When you expect a large number of errors but do not want them reported.
- To manage the size of the discard file.

More than one instance of `MAXDISCARDRECS` can be used in a parameter file to specify different maximums for different sets of `MAP` statements. An instance of `MAXDISCARDRECS` applies to all subsequent `MAP` statements until the next instance of `MAXDISCARDRECS` is encountered. The minimum is 0.

Default

None

Syntax

```
MAXDISCARDRECS number
```

number
The maximum number of errors to report.

Example

```
MAXDISCARDRECS 1000
```

2.103 MAXGROUPS

Valid For

GLOBALS

Description

Use the MAXGROUPS parameter to specify the maximum number of process groups that can run in an instance of Oracle GoldenGate. The Manager process checks this parameter to determine its resource allocations. The GGSCI process checks this parameter to control the maximum number of groups that it allows to be created.

Each Replicat thread in a coordinated Replicat group is considered to be a *group* in the context of MAXGROUPS. Therefore, the value of the MAXTHREADS option of COORDINATED in the ADD REPLICAT command (default is 25), plus the number of other Replicat and Extract groups in the Oracle GoldenGate instance, cannot exceed the MAXGROUPS value, or ADD REPLICAT returns an error.

The actual number of processes that can run on a given system depends on the system resources that are available. If those resources are exceeded, Oracle GoldenGate returns errors regardless of the setting of MAXGROUPS.

Default

1000 groups

Syntax

```
MAXGROUPS number
```

number
The number of groups allowed in one instance of Oracle GoldenGate. Valid values are from 1000 to 5000.

Example

```
MAXGROUPS 1500
```

2.104 MAXSQLSTATEMENTS

Valid For

Replicat

Description

Use the MAXSQLSTATEMENTS parameter to control the number of prepared SQL statements that can be used by Replicat both in regular processing mode and in BATCHSQL mode. The value for MAXSQLSTATEMENTS determines the number of open

cursors that Replicat maintains. Make certain that the database can support the specified number of cursors, plus the cursors that other applications and processes use. Before changing MAXSQLSTATEMENTS, contact Oracle Support.

When setting MAXSQLSTATEMENTS for a coordinated Replicat, take into account that the specified maximum number of cursors is applied to each thread in the configuration, not as an aggregate threshold for Replicat as a whole. For example, if MAXSQLSTATEMENTS 100 is specified, it is possible for each thread to have 99 open cursors without any warning or error from Replicat.

See "BATCHSQL" for more information about BATCHSQL mode.

Default

250 cursors

Syntax

```
MAXSQLSTATEMENTS number
```

number
The maximum number of cursors that Replicat (or each thread in a coordinated Replicat) can use. Valid values are from 1 to 250.

Example

```
MAXSQLSTATEMENTS 200
```

2.105 MAXTRANSOPS

Valid For

Replicat (Not supported in integrated Replicat mode)

Description

Use the MAXTRANSOPS parameter to split large source transactions into smaller ones on the target system. This parameter can be used when the target database is not configured to accommodate large transactions. For example, if the Oracle rollback segments are not large enough on the target to reproduce a source transaction that performs one million deletes, you could specify MAXTRANSOPS 10000, which forces Replicat to issue a commit after each group of 10,000 deletes.

To use MAXTRANSOPS is to alter the transactional boundaries that are imposed by the source application, even though Replicat applies the operations in the correct order. This can cause errors if Extract fails during that transaction. Extract rewrites the transaction to the end of the trail, instead of overwriting the old one. Because the trail is sequential, Replicat starts processing the old transaction and must roll it back when it receives the recovery marker and the new transaction, and then start applying the new transaction. If MAXTRANSOPS caused Replicat to split the original transaction into multiple smaller transactions, Replicat may only be able to roll back the portion that was not committed to the target. When Replicat processes the committed operations again, they will result in duplicate-row errors or missing-row errors, depending on the SQL operation type. The minimum is 1.

> **Note:**
>
> When troubleshooting Replicat abend errors, Oracle Support may request GROUPTRANSOPS to be set to 1 and MAXTRANSOPS to be set to 1. This is only a temporary configuration for troubleshooting purposes and should not be used permanently in production, or it will cause data integrity errors.

Default

10,000,000

Syntax

```
MAXTRANSOPS number
```

number
The number of operations to portion into a single transaction group.

Example

```
MAXTRANSOPS 10000
```

2.106 MGRSERVNAME

Valid For

GLOBALS

Description

Use the MGRSERVNAME parameter in a GLOBALS parameter file to specify the name of the Manager process when it is installed as a Windows service. This parameter is only required when installing multiple instances of Manager as a service on the same system, for example when installing multiple Oracle GoldenGate instances or when also installing the Oracle GoldenGate Veridata Agent, which uses a Manager process.

There must be a GLOBALS file containing MGRSERVNAME for each Manager service that is installed with the INSTALL utility. The files must be created before the services are installed, because the installation program refers to MGRSERVNAME when registering the service name on the system.

Default

None

Syntax

```
MGRSERVNAME name
```

name
A one-word name for the Manager service.

Example

```
MGRSERVNAME Goldengate
```

2.107 NAMECCSID

Valid for

GLOBALS, Extract, Replicat, DEFGEN for DB2 on IBM i

Description

Use the `NAMECCSID` parameter to specify the CCSID (coded character set identifier) of the database object names stored in the SQL catalog tables. The SQL catalog tables are created with the CCSID of the system, but the actual database object names could be represented in the catalog with characters from a different CCSID. The catalog does not indicate this difference when queried, and therefore Oracle GoldenGate could retrieve the name incorrectly unless `NAMECCSID` is present to supply the correct CCSID value.

To set the CCSID for a GGSCI session, use the `SET NAMECCSID` command.

To view the current CCSID, use the `SHOW` command. If the CCSID is not set through the GGSCI session or through the parameter `NAMECCSID`, the `SHOW` value will be `DEFAULT`.

Default

```
DEFAULT
```

Syntax

```
NAMECCSID {CCSID | DEFAULT}
```

CCSID
A valid DB2 for i coded character set identifier that is to be used for object names in catalog queries.

DEFAULT
Indicates that the system CCSID is to be used for object names in catalog queries.

Example

```
NAMECCSID 1141
```

2.108 NAMEMATCH parameters

Valid For

GLOBALS

Description

Use the `NAMEMATCH` parameters to control the behavior of fallback name mapping. Fallback name mapping is enabled by default when the source database is case-sensitive and the target database support both case-sensitive and case-insensitive object names, such as Oracle and DB2 LUW.

By default, `NAMEMATCHIGNORECASE` fallback name matching works as follows: When a source table name is case-sensitive, Oracle GoldenGate applies case-sensitive

wildcard mapping on the target database to find an exact match. If the target database does not contain the exact target table name, including case, fallback name mapping performs a case-insensitive target table mapping to find a name match.

Default

`NAMEMATCHIGNORECASE`

Syntax

```
NAMEMATCHIGNORECASE | NAMEMATCHNOWARNING | NAMEMATCHEXACT
```

`NAMEMATCHIGNORECASE`
Performs a case-insensitive target table mapping to find a name match when the target database does not contain the exact target table name, including case.

`NAMEMATCHNOWARNING`
Outputs a warning message to the report file when fallback name matching is used.

`NAMEMATCHEXACT`
Disables fallback name mapping. If an exact, case-sensitive match is not found, Oracle GoldenGate returns an error and abends.

2.109 NOCATALOG

Valid For

DEFGEN

Description

Use `NOCATALOG` in the DEFGEN parameter file to remove the Oracle Database container name from table names before their definitions are written to the definitions file. This parameter is valid if the database supports container names or catalog names and the `DEFSFILE` parameter includes the `FORMAT RELEASE` option set to 12.1. Use this parameter if the definitions file is to be used for mapping to a database that only supports two-part names (*`owner.object`*).

DEFGEN abends with an error if duplicate *`schema.table`* names are encountered once the container or catalog names are removed. This prevents the possibility of processing errors caused by different sets of metadata having the same *`schema.table`* name when there is no catalog name to differentiate them.

Default

None

Syntax

```
NOCATALOG
```

2.110 NODUPMSGSUPPRESSION

Valid For

GLOBALS

Description

Use NODUPMSGSUPPRESSION to prevent the automatic suppression of duplicate informational and warning messages in the report file, the error log, and the system log files. A message is issued to indicate how many times a message was repeated.

Default

Automatically suppress duplicate messages.

Syntax

```
NODUPMSGSUPPRESSION
```

2.111 NUMFILES

Valid For

Extract and Replicat

Description

Use the NUMFILES parameter to control the initial number of memory structures that are allocated to contain information about tables specified in TABLE or MAP statements. NUMFILES must occur before any TABLE or MAP entries, and before the SOURCEDEFS or TARGETDEFS parameter, to have any effect.

When setting NUMFILES for a coordinated Replicat, take into account that the specified value is applied to each thread in the configuration, not as an aggregate threshold for Replicat as a whole. For example, if NUMFILES 500 is specified, it is possible for each thread to have 499 initial memory structures without any warning or error from Replicat.

To control the number of additional memory structures that are allocated dynamically once the NUMFILES value is reached, use the ALLOCFILES parameter. See "ALLOCFILES" for more information. The default values should be sufficient for both NUMFILES and ALLOCFILES, because memory is allocated by the process as needed, system resources permitting. The minimum is 1 and the maximum is 20000.

Default

1000

Syntax

```
NUMFILES number
```

number
The initial number of memory structures to be allocated. Do not set NUMFILES to an arbitrarily high number, or memory will be consumed unnecessarily. The memory of Oracle GoldenGate supports up to two million tables.

Example

```
NUMFILES 4000
```

2.112 OBEY

Valid For

Extract and Replicat

Description

Use the OBEY parameter to retrieve parameter settings from a file other than the current parameter file.

To use OBEY, create and save a parameter file that contains the parameters that you want to retrieve. This is known as an OBEY file. You can create a library of OBEY files that contain different, frequently used parameter settings. Then, use the OBEY parameter in the active parameter file to invoke the parameters in the OBEY file.

Upon encountering an OBEY parameter in the active parameter file, Oracle GoldenGate processes the parameters from the OBEY file and then returns to the active parameter file to process any remaining parameters.

OBEY statements cannot be nested within other OBEY statements.

Instead of using OBEY, or in addition to it, you can use Oracle GoldenGate macros to retrieve frequently used parameters. For more information about using macros, see the *Administering Oracle GoldenGate*.

Default

None

Syntax

```
OBEY file
```

file
The relative or fully qualified name of the file from which to retrieve parameters or commands.

Example

```
OBEY /home/ogg/myparams
```

2.113 OUTPUTFILEUMASK

Valid For

GLOBALS

Description

Use the OUTPUTFILEUMASK parameter to specify an octal umask for Oracle GoldenGate processes to use when creating all files. OUTPUTFILEUMASK is not valid for Windows systems.

Default

Umask of 027 (all privileges)

Syntax

```
OUTPUTFILEUMASK umask
```

umask
The umask value. Must be between 0 and 077; otherwise there will be an error: `Missing or invalid option for OUTPUTFILEUMASK.`

Example

```
OUTPUTFILEUMASK 066
```

2.114 OUTPUTFORMAT

Valid For

Extract

Description

Use the `OUTPUTFORMAT` parameter to output data in text, SQL, and XML formats.

Default

None

Syntax

```
OUTPUTFORMAT format_type  [, option]  [, ...]
```

OUTPUTFORMAT TEXT

Use the `TEXT` *`format_type`* to output data in external text format instead of the default Oracle GoldenGate canonical format. You can format output that is compatible with most database load utilities and other programs that require ASCII input. This parameter is required by the *file-to-database-utility* initial load method.

This type of statement affects all extract files or trails that are listed after it in the parameter file. The relative order of the statements in the parameter file is important. If listed after a file or trail specification, `OUTPUTFORMAT TEXT` will not take effect.

`option` can be *one* of the following:

`INCLUDE (HEARTBEAT)`
Includes the heartbeat table records. By default, the heartbeat table records are ignored.

`BCP`
Formats the output for compatibility with SQL Server's Bulk Copy Program and other bulk load utilities.
The following options are ignored when the `BCP` option is specified:

- NAMES | NONAMES — Specifies whether or not to include column names. NAMES is the default.
- NULLISSPACE — Output NULL columns as empty columns. Without NULLISSPACE, NULL columns are output as NULL.
- PLACEHOLDERS — Outputs placeholder for missing columns.
- NOHDRFIELDS — Does not include any metadata, such as the before and after indicator, and transaction information. Outputs column data only.
- DELIMITER *'delimiter'* — Specifies the field delimiter character. To specify tabulation, use TAB. The default is a comma ‘,’.
- OP | _NOOP — Specifies whether or not to include operation type indicator (I, D, U, V). OP is the default.
- IND | _NOIND — Specifies whether or not to include the before and after image indicator (B or A). IND is the default.
- _TRANSTMTS | _NOTRANSTMTS — Specifies whether or not to include transaction information. _TRANSTMTS is the default.
- _WHOLEFILE — Includes the fully-qualified object name including the schema name.
- _FILE — Includes the object name only.

SQLLOADER
Produces a fixed-length text formatted file that is compatible with the Oracle SQL*Loader utility or the IBM load utility.

DATE | TIME | TS
Specifies the record timestamp precision to output. By default, this parameter does not output record timestamp. You can use one of the following:

- DATE outputs the date (year to day).
- TIME outputs the time (year to second).
- TS outputs the transaction timestamp (year to microseconds).

SQLLOADER
Produces a fixed-length, ASCII-formatted file that is compatible with the Oracle SQL*Loader utility or the IBM Load Utility program.

OUTPUTFORMAT SQL

Use the OUTPUTFORMAT SQL parameter to output data in external SQL format, instead of the default Oracle GoldenGate canonical format. OUTPUTFORMAT SQL generates SQL statements (INSERT, UPDATE, DELETE) that can be applied to SQL tables by utilities other than Oracle GoldenGate Replicat.

INCLUDE (HEARTBEAT)
Includes the heartbeat table records. By default, heartbeat table records are ignored.

ENCODING *encoding*
Outputs the SQL format file in the specified encoding. Oracle GoldenGate character set names are supported. By default, is current operating system character set. No character set conversion on column data is performed with the default character set.

The following options specify the specific output format. The options are exclusive so cannot be specified together.

ORACLE
Formats records for compatibility with Oracle Databases by converting date and time columns to a format accepted by SQL*Plus.

SQLPLUS
Formats records for compatibility with Oracle Databases by converting date and time columns to a format accepted by SQL*Plus.

SQLLOADER
Produces a fixed-length text formatted file that is compatible with the Oracle SQL*Loader utility or the IBM load utility program.
This is exactly the same as `OUTPUTFORMAT TEXT SQLLOADER`, which Oracle recommends that you use..

TRANSTMTS | NOTRANSTMTS
Includes SQL transaction information as comment. `NOTRANSTMTS` is the default. For example:

```
–B, 2016-07-09:09:9:21.000000,1357991461,627
```

WHOLEFILE
Includes the fully-qualified object name including the schema name.

FILE
Includes the object name only.

NOPKUPDATES
Formats `PKUPDATE` and `UNIFIED UPDATE` operations as a pair of `DELETE` and `INSERT` operations. `PKUPDATE` and `UNIFIED UPDATE` operations are formatted as an `UPDATE` operation if the option is not specified. This option is ignored if the `SQLLOADER` option is used.

OUTPUTFORMAT XML

Use the `OUTPUTFORMAT XML` parameter to output data in XML format, instead of the default Oracle GoldenGate canonical format. An `OUTPUTFORMAT XML` statement affects all Extract files or trails that are defined after it. By default, the XML is output in the character set of the local operating system.

XML stored as CLOB or BLOB is output up to 4000 bytes. To include larger XML stored as BLOB or CLOB, use the `ENCODING` option.

XML stored as CLOB is always output in a `CDATA` section regardless of its size. This is to avoid the overhead of converting reserved characters such as <, > and & to the appropriate XML representation.

Binary data including BLOB are encoded as Base64, which represents binary data in an ASCII string format and allows output to XML.

The XML, the database object names, such as table and column names, and `CHAR` and `VARCHAR` data are written in the default character set of the operating system unless the `ENCODING` option is used to output in UTF-8.

INCLUDE (HEARTBEAT | LOB | USERTOKEN)
Includes the heartbeat table records. LOB more than 4000 bytes and Oracle GoldenGate user tokens.
By default, heartbeat table records are ignored and doesn't include LOB more than 4000 bytes and user tokens.
BLOB more than 4000 bytes is encoded in Base64, and CLOB more than 4000 bytes is formatted in a `CDATA` section.

INLINEPROPERTIES | NOINLINEPROPERTIES
Controls whether or not properties are included within the XML tag or written separately. `INLINEPROPERTIES` is the default.

TRANS | NOTRANS
Controls whether or not transaction boundaries and commit timestamps should be included in the XML output. `TRANS` is the default.

CLOSETRANS | NOCLOSETRANS
Forces the closure of opened transaction boundaries and commits the timestamp upon rollover. It adds same transaction boundaries and commit timestamp tags to the next XML file after rollover.
The option is ignored if the `TRANS` option is not specified.

ENCODING xml_encoding
Outputs an XML file in the specified encoding. The default is UTF-8. The following MIME encoding names are supported.

UTF-8	ISO-10646 UTF-8, surrogate pairs are 4 bytes per character
UTF-16	ISO-10646 UTF-16
windows-1250	Windows Central Europe
windows-1251	Windows Cyrillic
windows-1252	Windows Latin-1
windows-1253	Windows Greek
windows-1254	Windows Turkish
windows-1255	Windows Hebrew
windows-1256	Windows Arabic
windows-1257	Windows Baltic
windows-1258	Windows Vietnam
windows-874	Windows Thai
IBM437	DOS Latin-1
IBM775	DOS 775, Baltic
IBM850	DOS multilingual
cp851	DOS Greek-1
IBM852	DOS Latin-2
IBM855	DOS Cyrillic
IBM857	DOS Turkish
IBM00858	DOS Multilingual with Euro
IBM860	DOS Portuguese
IBM861	DOS Icelandic
IBM862	DOS Hebrew
IBM863	DOS French
IBM864	DOS Arabic
IBM865	DOS Nordic
IBM866	DOS Cyrillic / GOST 19768-87
IBM868	DOS Urdu
IBM869	DOS Greek-2
ISO-8859-1	ISO-8859-1 Latin-1/Western Europe

SO-8859-2	ISO-8859-2 Latin-2/Eastern Europe
ISO-8859-3	ISO-8859-3 Latin-3/South Europe
ISO-8859-4	ISO-8859-4 Latin-4/North Europe
ISO-8859-5	ISO-8859-5 Latin/Cyrillic
ISO-8859-6	ISO-8859-6 Latin/Arabic
ISO-8859-7	ISO-8859-7 Latin/Greek
ISO-8859-8	ISO-8859-8 Latin/Hebrew
ISO-8859-9	ISO-8859-9 Latin-5/Turkish
ISO-8859-10	ISO-8859-10 Latin-6/Nordic
ISO-8859-13	ISO-8859-13 Latin-7/Baltic Rim
ISO-8859-14	ISO-8859-14 Latin-8/Celtic
ISO-8859-15	ISO-8859-15 Latin-9/Western Europe
ISO-8859-16	ISO-8859-16, Latin-10, South Eastern Europe
KOI8-R	KOI8-R, Russian
KOI8U	KOI8-U, Ukranian
TIS-620	Thai Industrial Standard 620-2533
DEC-MCS	DEC Multilingual
hp-roman8	HP Latin-1 Roman8
Shift_JIS	Shift_JIS, Windows-932
GBK	GBK, Windows-936
KSC_5601	KSC-5601, Windows-949
Big5	Big-5 Traditional Chinese, Windows-950
EUC-JP	EUC Japanese
GB2312	GB-2312-1980
EUC-KR	EUC Korean
GB18030	GB-18030
HZ-GB-2312	HZ GB-2312
Big5-HKSCS	Big-5, HongKong extension

Example

```
FORMATXML NOINLINEPROPERTIES, NOTRANS
```

2.115 OVERRIDEDUPS | NOOVERRIDEDUPS

Valid For

Replicat

Description

Use the `OVERRIDEDUPS` and `NOOVERRIDEDUPS` parameters to control whether or not Replicat overwrites an existing record in the target database with a replicated one if both records have the same key.

- `OVERRIDEDUPS` overwrites the existing record. It can be used for initial loads where you do not want to truncate target tables prior to the load, or for the resynchronization of a target table with a trusted source. Use the `SQLDUPERR` parameter with `OVERRIDEUPS` to specify the numeric error code that is returned by the database for duplicate `INSERT` operations. See "SQLDUPERR" for more information.
- `NOOVERRIDEDUPS`, the default, generates a duplicate-record error instead of overwriting the existing record. You can use an exceptions `MAP` statement with a `SQLEXEC` clause to initiate a response to the error. Otherwise, the transaction may

abend. For more information about exceptions maps, see *Administering Oracle GoldenGate*.

To bypass duplicate records without causing Replicat to abend when an exceptions map is not available, specify a REPERROR parameter statement similar to the following, where *error* is the database error number for primary key constraint errors.

```
REPERROR (error, IGNORE)
```

For example, the statement for an Oracle database would be:

```
REPERROR (1, IGNORE)
```

Replicat writes ignored duplicate records to the discard file.

Place OVERRIDEDUPS or NOOVERRIDEDUPS before the TABLE or MAP statements that you want it to affect. You can create different rules for different groups of TABLE or MAP statements. The parameters act as toggles: one remains in effect for subsequent TABLE or MAP statements until the other is encountered.

OVERRIDEDUPS is enabled automatically when HANDLECOLLISIONS is used. See "HANDLECOLLISIONS | NOHANDLECOLLISIONS" for more information.

> **WARNING:**
>
> When OVERRIDEDUPS is in effect, records might not be processed in chronological order across multiple Replicat processes.

Default

```
NOOVERRIDEDUPS
```

Syntax

```
OVERRIDEDUPS | NOOVERRIDEDUPS
```

2.116 PTKCAPTUREPROCSTATS

Valid For

Extract, Replicat, and Manager

Description

Use PTKCAPTUREPROCSTATS enables the capture of process and thread statistics for the PTK Monitoring.

Default

```
true
```

Syntax

```
PTKCAPTUREPROCSTATS seconds
```

capture
Controls whether or not PTK Monitoring statistics are captured with either `true` or `false`.

Examples

```
PTKCAPTUREPROCSTATS false
```

2.117 PTKMONITORFREQUENCY

Valid For

Extract, Replicat, and Manager

Description

Use `PTKMONITORFREQUENCY` to set the monitoring collection frequency interval.

Default

One second.

Syntax

```
PTKMONITORFREQUENCY seconds
```

seconds
Specifies the time interval, in seconds, for monitoring collection to occur. The minimum is 1 seconds and the maximum is 60 seconds.

Examples

```
PTKMONITORFREQUENCY 10
```

2.118 PRESERVETARGETTIMEZONE

Valid For

Replicat

Description

Use the `PRESERVETARGETTIMEZONE` parameter to override the default Replicat session time zone. By default, Replicat sets its session to the time zone of the source database, as written to the trail by Extract. `PRESERVETARGETTIMEZONE` causes Replicat to set its session to the time zone of the target database.

Default

None

Syntax

```
PRESERVETARGETTIMEZONE
```

2.119 PROCEDURE

This is an option that can be specified as a stand-alone statement in extract and replicat parameter file. It indicates which feature group of procedural calls will be replicated.

Syntax

```
PROCEDURE [ INCLUDE | EXCLUDE ]  FEATURE [ALL_SUPPORTED | feature_list]
```

Examples

Example 1
Include all system supplied packages:

```
PROCEDURE INCLUDE FEATURE ALL_SUPPORTED
```

Example 2
Include specific packages

```
PROCEDURE INCLUDE FEATURE AQ, FGA, DBFS
```

Example 3
Exclude a specific packages

```
PROCEDURE INCLUDE FEATURE ALL_SUPPORTED
PROCEDURE EXCLUDE FEATURE REDFINITION
```

2.120 RECOVERYOPTIONS

Valid For

Extract

Description

Use the `RECOVERYOPTIONS` parameter to control how Extract handles the re-writing of data to the trail after it fails while in the process of writing transaction data.

Parameter Dependencies

There is a dependency between the `RECOVERYOPTIONS` parameter and the `FORMAT` option of `EXTFILE`, `EXTTRAIL`, `RMTFILE`, and `RMTTRAIL`.

Default

None

Syntax

```
RECOVERYOPTIONS
```

Example

```
RECOVERYOPTIONS
```

2.121 REPERROR

Valid For

Replicat

Description

Use the REPERROR parameter to control how Replicat responds to errors. The default response of Replicat to any error is to abend.

You can use one REPERROR statement to handle most errors in a default manner, while using one or more other REPERROR statements to handle specific errors differently. For example, you can ignore duplicate-record errors but abend processing in all other cases.

You can use REPERROR globally (at the root of the parameter file) to affect all MAP statements that follow it, or you can use it within a MAP statement to affect the tables specified in that statement. Using REPERROR within a MAP statement gives you the ability to handle errors in a particular way for each thread of a coordinated Replicat.

Using Record-level Error Handling

All REPERROR options except TRANSDISCARD and TRANSEXCEPTION apply an error-handling action in response to an individual SQL operation on an individual record. Other, error-free records in the same transaction are processed as configured in the MAP statements and other parameters in the parameter file, as applicable.

Using Transaction-level Error Handling

The TRANSDISCARD, TRANSEXCEPTION, and ABEND options apply an error-handling action to an entire transaction. The triggering error can occur on an individual record in the transaction or on the commit operation. (Commit errors do not have a particular record associated with them.) These options can be used to:

- prevent an entire source transaction from being replicated to the target when any error is associated with it.
- respond to a commit error when deferred constraint checking is enabled on the target.

TRANSDISCARD and TRANSEXCEPTION are mutually exclusive.

Effect of Other Parameters on Transaction-level Options

TRANSDISCARD and TRANSEXCEPTION honor the boundaries of the source transaction; however, the presence of BATCHSQL, GROUPTRANSOPS, or MAXTRANSOPS in the parameter file may affect the error-handling logic or outcome, because they alter transaction boundaries.

Effect of BATCHSQL and GROUPTRANSOPS

BATCHSQL or GROUPTRANSOPS (the default) both group SQL operations from different transactions into larger transactions to improve performance, while maintaining transactional order. When these parameters are in effect and any error occurs, Replicat first tries to resolve it by entering an alternate processing mode (see the documentation for those parameters). If the error persists, TRANSDISCARD or

TRANSEXCEPTION comes into effect, and Replicat reverts to source-processing mode as follows:

1. It rolls back the grouped or arrayed transaction.
2. It replays the offending transaction one SQL operation at a time, using the same transaction boundaries as the source transaction.
3. It performs the discard logic (TRANSDISCARD) or exceptions-mapping (TRANSEXCEPTION). (See those option descriptions for more detail.)
4. It resumes BATCHSQL or GROUPTRANSOPS mode after the TRANSDISCARD error handling is completed.

Effect of MAXTRANSOPS

The integrity of TRANSDISCARD and TRANSEXCEPTION transaction-level error handling can be adversely affected by the setting of the MAXTRANSOPS parameter. MAXTRANSOPS causes Replicat to split very large replicated source transactions into smaller transactions when it applies them on the target.

The TRANSDISCARD and TRANSEXCEPTION logic cause Replicat to roll back to the first record after the last successful commit. This may or may not be the actual beginning of the offending transaction. It depends on whether that transaction was split up and parts of it are in the previously committed transactions. If that is the case, Replicat cannot apply the TRANSDISCARD or TRANSEXCEPTION action to the whole transaction as it was issued on the source, but only to the part that was rolled back from the target.

If you use MAXTRANSOPS, make certain that it is set to a value that is larger than the largest transaction that you expect to be handled by TRANSDISCARD and TRANSEXCEPTION. This will ensure that transactions are not be split apart into smaller ones on the target.

Effect of Transaction-level Options on Statistics

The output of informational commands in GGSCI, such as STATS REPLICAT, will show the total number of records in the transaction that was processed by TRANSDISCARD or TRANSEXCEPTION logic. This number may reflect the following:

- Replicat writes all records of the transaction to the discard file, including any records that were excluded from Oracle GoldenGate processing by means of a FILTER or WHERE clause in a MAP statement.
- If a source table in the transaction has multiple targets, the discarded transaction will contain multiple copies of each record, one for each target.
- Replicat ignores any exceptions mapping statements (as specified with EXCEPTIONSONLY or MAPEXCEPTION in a MAP statement) when discarding the transaction.

Replicat abends on errors that are caused by the discard processing (TRANSDISCARD) or exceptions mapping (TRANSEXCEPTION).

Getting More Information about Error Handling

See *Administering Oracle GoldenGate* for more information about configuring error handling.

See "TABLE | MAP" for more information about the MAP parameter.

Default

TRANSABORT for deadlocks; ABEND for all others

Syntax

```
REPERROR {
(
{DEFAULT | DEFAULT2 | SQL_error | user_defined_error},
{ABEND |
DISCARD |
EXCEPTION |
IGNORE |
RETRYOP [MAXRETRIES n] |
TRANSABORT [, MAXRETRIES] [, DELAYSECS n | DELAYCSECS n] |
TRANSDISCARD |
TRANSEXCEPTION
}
{PROCEDURE,[ABEND|IGNORE|DISCARD]}) |
RESET }
```

Error Specification Options

DEFAULT
Sets a global response to all errors except those for which explicit REPERROR statements are specified.

DEFAULT2
Provides a backup default action when the response for DEFAULT is set to EXCEPTION. Use DEFAULT2 when an exceptions MAP statement is not specified for a MAP statement for which errors are anticipated.

SQL_error
A SQL error number. This can be a record-level error or a commit-level error if using TRANSDISCARD and TRANSEXCEPTION.

user_defined_error
A user-defined error that is specified with the RAISEERROR option of a FILTER clause within a MAP statement.

Error Response Options

ABEND
Rolls back the transaction and terminates processing abnormally. ABEND is the default.

DISCARD
Logs the offending operation to the discard file but continue processing the transaction and subsequent transactions.

EXCEPTION
Handles the operation that causes an error as an exception, but processes error-free operations in the transaction normally. Use this option in conjunction with an exceptions MAP statement or to work with the MAPEXCEPTION option of MAP. For example, you can map columns from failed update statements into a "missing updates" table. In the parameter file, specify the exceptions MAP statement after the MAP statement for which the error is anticipated.

EXCEPTION applies exception handling only to an individual SQL operation on an individual record. To apply exception handling to the entire transaction, use the TRANSEXCEPTION option.

> **Note:**
>
> When the Conflict Detection and Resolution (CDR) feature is active, CDR automatically treats all operations that cause errors as exceptions if an exceptions MAP statement exists for the affected table. In this case, REPERROR with EXCEPTION is not necessary, but you should use REPERROR with other options to handle conflicts that CDR cannot resolve, or for conflicts that you do not want CDR to handle.

IGNORE
Ignores the error.

RETRYOP [MAXRETRIES *n*]
Retries the offending operation. Use the MAXRETRIES option to control the number of retries. For example, if a 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.

TRANSABORT [, MAXRETRIES *n*] [, DELAYSECS *n* | DELAYCSECS *n*]
Aborts the transaction and repositions to the beginning of the transaction. This sequence continues either until the record(s) are processed successfully or MAXRETRIES expires. If MAXRETRIES is not set, the TRANSABORT action will loop continuously.
Use one of the DELAY options to delay the retry. DELAYSECS *n* sets the delay in seconds and the default is 60 seconds. DELAYCSECS *n* sets the delay in centiseconds.
The TRANSABORT option is useful for handling timeouts and deadlocks on databases that support those conditions.

TRANSDISCARD
Discards the entire source transaction if any operation within that transaction, including the commit operation, causes a Replicat error that is listed in the REPERROR error specification. Replicat aborts the transaction and, if the error occurred on a record, writes that record to the discard file. Replicat then replays the transaction and writes all of the records to the discard file, including the commit record. Replicat abends on errors that are caused by the discard processing.
If the discarded record has already been data-mapped to a target record, Replicat writes it to the discard file in the target format; otherwise, it will be written in source format. The replayed transaction itself is always written in source format.
TRANSDISCARD supports record-level errors as well as commit errors.
Additional information is at the beginning of this topic.

TRANSEXCEPTION
If an error specified with REPERROR occurs on any record in a transaction, performs exceptions mapping for every record in the transaction according to its corresponding exceptions-mapping specification, as defined by a MAPEXCEPTION or EXCEPTIONSONLY clause in an exceptions MAP statement. If any record does not have a corresponding exceptions mapping specification, or if there is an error writing to the exceptions table, Replicat abends with an error message.

When an error is encountered and TRANSEXCEPTION is being used, Replicat aborts the transaction and, if the error occurred on a record, writes that record to the discard file. Replicat replays the transaction and examines the source records to find the exceptions-mapping specifications, and then executes them.
TRANSEXCEPTION supports record-level errors as well as commit errors. To handle errors at the record level (for individual SQL operations), without affecting error-free operations in the same transaction, use the EXCEPTION option in a MAP statement.

PROCEDURE,[ABEND|IGNORE|DISCARD]
Use PROCEDURE to configure behavior of Replicat when a procedural replication error occurs. By default, Replicat will ABEND when a procedural replication error occurs. The IGNORE option ignores the call that failed. The DISCARD option stages the discarded errors in the apply error queue in the target database. These errors can be re-executed or deleted at a later time.

RESET
Use a REPERROR RESET statement to remove error-handling rules specified in previous REPERROR parameters and apply default error handling to all MAP statements that follow.

Examples of Using REPERROR Globally

These examples show REPERROR as used at the root of the parameter file to set global error-handling rules. You can override any or all of these rules for any given table or tables by using REPERROR in a MAP statement. See "Examples of Using REPERROR Globally and in a MAP Statement".

Example 1
The following example demonstrates how to stop processing for most errors, but ignore duplicate-record errors.

```
REPERROR (DEFAULT, ABEND)
REPERROR (-1, IGNORE)
```

Example 2
The following example invokes an exceptions MAP statement created to handle errors on the account table. Errors on the product table cause Replicat to end abnormally because an exceptions MAP statement was not defined.

```
REPERROR (DEFAULT, EXCEPTION)
REPERROR (DEFAULT2, ABEND)
MAP sales.product, TARGET sales.product;
MAP sales.account, TARGET sales.account;
INSERTALLRECORDS
MAP sales.account, TARGET sales.account_exception,
EXCEPTIONSONLY,
COLMAP (account_no = account_no,
optype = @GETENV ('lasterr', 'optype'),
dberr = @GETENV ('lasterr', 'dberrnum'),
dberrmsg = @GETENV ('lasterr', 'dberrmsg'));
```

Example 3
The following applies error rules for the first MAP statement and then restores the default of ABEND to the second one.

```
REPERROR (-1, IGNORE)
MAP sales.product, TARGET sales.product;
```

```
REPERROR RESET
MAP sales.account, TARGET sales.account;
```

Example 4
The following discards the offending record and then replays the entire transaction if any operation on a record within it generates an error 1403. Other error types cause Replicat to abend.

```
REPERROR DEFAULT ABEND
REPERROR 1403 TRANSDISCARD
```

Example 5
The following discards the offending record and then replays the entire transaction to search for an exceptions-mapping specification that writes to the exceptions table that is named `tgtexception`. Other errors cause Replicat to discard the offending record (if applicable) and then abend.

```
REPERROR DEFAULT ABEND
REPERROR 1403 TRANSEXCEPTION
MAP src, TARGET tgt, &
MAPEXCEPTION (TARGET tgtexception, INSERTALLRECORDS, COLMAP (…) );
```

Examples of Using REPERROR Globally and in a MAP Statement

The following examples show different ways that `REPERROR` can be used in a `MAP` statement in conjunction with a global `REPERROR` statement.

Example 1

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

In the preceding 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.

Example 2

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

In the preceding 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).

Example 3

```
REPLICAT group_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);
```

In the preceding 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.

Example 4

```
REPERROR DEFAULT ABEND
REPERROR 1403 TRANSDISCARD.
MAP src, TARGET tgt, REPERROR(600 TRANSDISCARD);
```

In the preceding example, if error 600 is encountered while applying source table src to target table tgt, the whole transaction is written to discard file. Encountering error 1403 also results in the same action based on the global REPERROR specification. On the other errors, the process simply discards only the offending record and then abends.

2.122 REPFETCHEDCOLOPTIONS

Valid For

Replicat

Description

Use the REPFETCHEDCOLOPTIONS parameter to determine how Replicat responds to operations for which a fetch from the source database was required. The Extract process fetches column data when the transaction record does not contain enough information to construct a SQL statement or when a FETCHCOLS clause is used. See "{FETCHCOLS | FETCHCOLSEXCEPT} (*column_list*)" for more information.

Default

None

Syntax

```
REPFETCHEDCOLOPTIONS
[, INCONSISTENTROW ALLOW|IGNORE|REPORT|DISCARD|ABEND]
[, LATESTROWVERSION ALLOW|IGNORE|REPORT|DISCARD|ABEND]
[, MISSINGROW ALLOW|IGNORE|REPORT|DISCARD|ABEND]
[, NOFETCH ALLOW|IGNORE|REPORT|DISCARD|ABEND]
[, REDUNDANTROW ALLOW|IGNORE|REPORT|DISCARD|ABEND]
[, SNAPSHOTROW ALLOW|IGNORE|REPORT|DISCARD|ABEND]
[, SETIFMISSING string]
```

INCONSISTENTROW
Determines the action to perform when column data was successfully fetched by row ID, but the key did not match. Either the row ID was recycled or a primary key update occurred after this operation (and prior to the fetch). Valid values are

ALLOW
Process the operation unless the record length is zero (0).

IGNORE
Ignore the condition and continue processing.

REPORT
Write the record to the discard file and process the operation.

DISCARD
Discard the data and do not process the row.

ABEND
Discard the data and quit processing.

LATESTROWVERSION ***action***
Provides a response when column data was fetched from the current row in the table. Valid values are:

ALLOW
Process the operation unless the record length is zero (0).

IGNORE
Ignore the condition and continue processing.

REPORT
Write the record to the discard file and process the operation.

DISCARD
Discard the data and do not process the row.

ABEND
Discard the data and quit processing.

NOFETCH ***action***
Prevents fetching. One use for this option is when the database is a standby and Oracle GoldenGate does not have a database connection. In this case, an attempt to fetch from the database would result an error. Other scenarios may warrant the use of this parameter as well.
When Oracle GoldenGate cannot fetch data it normally would fetch, it probably will cause data integrity issues on the target.
The following are valid actions that can be taken when a `NOFETCH` is encountered:

ABEND
Write the operation to the discard file and abend the Replicat process. This is the default.

ALLOW
Process the operation unless the record length is zero (0).

IGNORE
Ignore the operation. If fetch statistics are being reported in the process report (based on `STATOPTIONS` settings) they will be updated with this result.

REPORT
Write the record to the discard file and process the operation.

DISCARD
Write the record to the discard file, but do not process the operation. If fetch statistics are being reported in the process report (based on STATOPTIONS settings) they will be updated with this result.

MISSINGROW *action*
Provides a response when only part of a row (the changed values) is available to Replicat for processing. The column data that is missing from the trail typically could not be fetched because the row was deleted between the time the change record was created and when the fetch was triggered, or because the row image required was older than the undo retention specification.
Valid values are:

ALLOW
Process the operation unless the record length is zero (0).

IGNORE
Ignore the condition and continue processing.

REPORT
Write the record to the discard file and process the operation.

DISCARD
Discard the data and do not process the partial row.

ABEND
Discard the data and quit processing.

REDUNDANTROW
Indicates that column data was not fetched because column data was previously fetched for this record.

SETIFMISSING [*string*]
Provides a value when a fetch was unsuccessful (and the value is missing from the trail record) but the target column has a not-null constraint. It takes an optional ASCII string as a value for CHAR and BINARY data types or defaults to the following.
CHAR, VARCHAR: Single space
BINARY, VARBINARY: A NULL byte
TIMESTAMP: Current date/time
FLOAT, INTEGER: Zero
Besides SETIFMISSING, you can use the COLMAP clause of the MAP statement to map a value for the target column. See "COLMAP *(column_mapping)*" for more information.

SNAPSHOTROW
Indicates that column data was fetched from a snapshot. Generally, this option would only be used for reporting or discarding operations. Valid values are:

ALLOW
Process the operation unless the record length is zero (0).

IGNORE
Ignore the condition and continue processing.

REPORT
Write the record to the discard file and process the operation.

DISCARD
Discard the data and do not process the row.

ABEND
Discard the data and quit processing.

2.123 REPLACEBADCHAR

Valid For

Extract and Replicat

Description

Use the REPLACEBADCHAR parameter to control the response of the process when a valid code point does not exist for either the source or target character set when mapping character-type columns. By default, the check for invalid code points is only performed when the source and target databases have different character sets, and the default response is to abend. You can use the FORCECHECK option to force the process to check for invalid code points when the source and target databases have the same character set. REPLACEBADCHAR applies globally.

Default

ABORT

Syntax

```
REPLACEBADCHAR {ABORT | SKIP | ESCAPE | SUBSTITUTE string | NULL | SPACE}
[FORCECHECK] [NOWARNING]
```

ABORT
The process abends on an invalid code point. This is the default.

SKIP
The process skips the record that has the invalid code point. Use this option with caution, because skipping a record can cause data discrepancies on the target.

ESCAPE
The process replaces the data value with an escaped version of the data value. Depending on the character set of the source database, the value is output as one of the following:

- If the source data is not UTF-16 (NCHAR/NVARCHAR), the output is hexadecimal (\xXX).
- If the source data is UTF-16, the output is Unicode (\uXXXX).

SUBSTITUTE *string*
The process replaces the data with a specified string, either Unicode notation or up to four characters. By default the default substitution character of the target character set is used for replacement.

NULL
The process replaces an invalid character with the value of NULL if the target column is nullable or, otherwise, assigns a white space (U+0020).

SPACE
The process replaces an invalid character with a white space (U+0020).

FORCECHECK
The process checks for invalid code points when the source and target databases have identical character sets. This overrides the default, where the validation is skipped when the source and target character sets are identical.

NOWARNING
The process suppresses warning messages related to conversion and validation errors.

Examples

Example 1
The following example replaces invalid code points with the value of `NULL`.

```
REPLACEBADCHAR NULL
```

Example 2
Because `ESCAPE` is specified, Oracle GoldenGate will replace the Euro symbol in a source `NCHAR` column with the escaped version of `u20AC`, because the target is ISO-8859-1, which does not support the Euro code point.

```
REPLACEBADCHAR ESCAPE
```

Example 3
The following substitutes a control character for invalid characters.

```
REPLACEBADCHAR SUBSTITUTE \u001A
```

2.124 REPLACEBADNUM

Valid For

Replicat

Description

Use the `REPLACEBADNUM` parameter to specify a substitution value for invalid numeric data encountered when mapping number columns. `REPLACEBADNUM` applies globally.

Default

Replace invalid numbers with `NULL`.

Syntax

```
REPLACEBADNUM {number | NULL | UNPRINTABLE}
```

number
Replace with the specified number.

NULL
Replace with `NULL` if the target column accepts `NULL` values; otherwise replace with zero.

UNPRINTABLE
Reject any column with unprintable data. The process stops and reports the bad value.

Examples

Example 1

```
REPLACEBADNUM 1
```

Example 2

```
REPLACEBADNUM NULL
```

2.125 REPLICAT

Valid For

Replicat

Description

Use the REPLICAT parameter to specify a Replicat group for online change synchronization. This parameter links the current run with previous runs, so that data changes are continually processed to maintain synchronization between source and target tables. Replicat will run continuously and maintain checkpoints in the data source and trail to ensure data integrity and fault tolerance throughout planned or unplanned process termination, system outages, or network failure.

Either REPLICAT or SPECIALRUN is required in the Replicat parameter file and must be the first entry. See "SPECIALRUN" for more information.

Default

None

Syntax

```
REPLICAT group_name
```

group_name
The group name as defined with the ADD REPLICAT command. To view the names of existing Replicat groups, use the INFO REPLICAT * command.

Example

```
REPLICAT finance
```

2.126 REPORT

Valid For

Extract and Replicat

Description

Use the REPORT parameter to specify the interval at which Extract or Replicat generates interim runtime statistics in a process report. The statistics are added to the existing report. By default, runtime statistics are displayed at the end of a run unless the process is intentionally killed.

The statistics for REPORT are carried over from the previous report. For example, if the process performed 10 million inserts one day and 20 million the next, and a report is generated at 3:00 each day, then the first report would show the first 10 million inserts, and the second report would show those plus the current day's 20 million inserts, totalling 30 million. To reset the statistics when a new report is generated, use the STATOPTIONS parameter with the RESETREPORTSTATS option. See "STATOPTIONS" for more information.

For more information about the process reports, see *Administering Oracle GoldenGate*.

Default

Generate runtime statistics at the end of each run.

Syntax

```
REPORT
{AT hh:mi |
ON day |
AT hh:mi ON day}
```

AT *hh:mi*
Generates the report at a specific time of the day. Using AT without ON generates a report at the specified time every day.

ON *day*
Generates the report on a specific day of the week. Valid values are:

```
SUNDAY
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
```

The values are not case-sensitive.

Examples

Example 1

```
REPORT AT 17:00
```

Example 2

```
REPORT ON SUNDAY AT 1:00
```

2.127 REPORTCOUNT

Valid For

Extract and Replicat

Description

Use the REPORTCOUNT parameter to report a count of transaction records that Extract or Replicat processed since startup. Each transaction record represents a logical database operation that was performed within a transaction that was captured by Oracle GoldenGate. The record count is printed to the report file and to the screen.

> **Note:**
>
> This count might differ from the number of records that are contained in the Oracle GoldenGate trail. If an operation affects data that is larger than 4K, it must be stored in more than one trail record. Hence, a report count might show 1,000 records (the database operations) but a trail count might show many more records than that. To obtain a count of the records in a trail, use the Logdump utility.

You can schedule record counts at regular intervals or after a specific number of records. Record counts are carried over from one report to the other.

REPORTCOUNT can be used only once in a parameter file. If there are multiple instances of REPORTCOUNT, Oracle GoldenGate uses the last one.

Default

None

Syntax

```
REPORTCOUNT [EVERY] count
{RECORD | RECORDS | SECOND | SECONDS | MINUTE | MINUTES | HOUR |HOURS} [, RATE]
```

count
The interval after which to output a count.

RECORD | RECORDS | SECOND | SECONDS | MINUTE | MINUTES | HOUR |HOURS
The unit of measure for *count*, in terms of records, seconds, minutes, or hours.

RATE
Reports the number of operations per second and the change in rate, as a measurement of performance. The Rate statistic is the total number of records divided by the total time elapsed since the process started. The Delta statistic is the number of records since the last report divided by the time since the last report.

> **Note:**
>
> The calculations are done using microsecond time granularity. The time intervals are shown without fractional seconds, and the rate values are shown as whole numbers.

Examples

Example 1
This example generates a record count every 5,000 records.

```
REPORTCOUNT EVERY 5000 RECORDS
```

Example 2
This example generates a record count every ten minutes and also reports processing statistics.

```
REPORTCOUNT EVERY 10 MINUTES, RATE
```

The processing statistics are similar to this:

```
12000 records processed as of 2011-01-01 12:27:40 (rate 203,delta 308)
```

2.128 REPORTROLLOVER

Valid For

Extract and Replicat

Description

Use the `REPORTROLLOVER` parameter to force report files to age on a regular schedule, in addition of when a process starts. For long or continuous runs, setting an aging schedule aids in controlling the size of the active report file and provides a more predictable set of archives that can be included in your archiving routine.

> **Note:**
>
> Report statistics are carried over from one report to the other. To reset the statistics in the new report, use the `STATOPTIONS` parameter with the `RESETREPORTSTATS` option.

You can specify a time of day, a day of the week, or both. Specifying just a time of day (`AT` option) without a day of the week (`ON` option) generates a report at the specified time every day.

Rollovers caused by this parameter do not generate runtime statistics in the process report:

- To control when runtime statistics are generated to report files, use the `REPORT` parameter.

- To generate new runtime statistics on demand, use the SEND EXTRACT or SEND REPLICAT command with the REPORT option.

Default

Roll reports at startup

Syntax

```
REPORTROLLOVER
{AT hh:mi |
ON day |
AT hh:mi ON day}
```

AT *hh:mi*
The time of day to age the file.
Valid values:

- hh is based on a 24-hour clock and accepts values of 1 through 23.
- mi accepts values from 00 through 59.

ON *day*
The day of the week to age the file. Valid values are:

```
SUNDAY
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
```

The values are not case-sensitive.

Examples

Example 1

```
REPORTROLLOVER AT 05:30
```

Example 2

```
REPORTROLLOVER ON friday
```

Example 3

```
REPORTROLLOVER AT 05:30 ON friday
```

2.129 REPOUDPPORT

Valid For

GLOBALS

Description

Use the REPOUDPPORT command to send metrics data to the Performance Metrics Server to monitor your data replications in the Classic Architecture.

followed by a port number.

Default

9330

Syntax

```
REPOUDPPORT port_number
```

port_number
Specifies the UDP multicast port that Oracle GoldenGate will use for monitoring . This parameter is optional. If more that one Oracle GoldenGate instance exists on a server, you must specify different port values.

Example

```
REPOUDPPORT 8080
```

2.130 RESTARTCOLLISIONS | NORESTARTCOLLISIONS

Valid For

Replicat

Description

Use the `RESTARTCOLLISIONS` and `NORESTARTCOLLISIONS` parameters to control whether or not Replicat applies `HANDLECOLLISIONS` logic after Oracle GoldenGate has stopped because of a conflict. By default, `NORESTARTCOLLISIONS` applies. However, there might be circumstances when you would want Oracle GoldenGate to apply `HANDLECOLLISIONS` logic for the first transaction after startup. For example, if the server is forcibly shut down, the database might have committed the last Replicat transaction, but Oracle GoldenGate might not have received the acknowledgement. Consequently, Replicat will retry the transaction upon startup. `HANDLECOLLISIONS` automatically handles the resultant errors that occur.

`RESTARTCOLLISIONS` enables `HANDLECOLLISIONS` functionality until the first Replicat checkpoint (transaction) is complete. You need not specify the `HANDLECOLLISIONS` parameter in the parameter file. After the first checkpoint, `HANDLECOLLISIONS` is automatically turned off.

See "HANDLECOLLISIONS | NOHANDLECOLLISIONS" for more information about handling collisions.

Default

```
NORESTARTCOLLISIONS
```

Syntax

```
RESTARTCOLLISIONS | NORESTARTCOLLISIONS
```

2.131 RMTFILE

Valid For

Extract

Description

Use the RMTFILE parameter to define the name of an extract file on a remote system to which extracted data will be written. Use this parameter for initial-load configurations. For online change synchronization, use the RMTTRAIL parameter.

The size of an extract file cannot exceed 2GB .

RMTFILE must be preceded by a RMTHOST statement, and it must precede any TABLE statements.

RMTFILE is deprecated and ignored for Data Pump.

You can encrypt the data in this file by using the ENCRYPTTRAIL parameter. See "ENCRYPTTRAIL | NOENCRYPTTRAIL" for more information.

Default

None

Syntax

```
RMTFILE file_name
[, APPEND]
[, PURGE]
[, MEGABYTES megabytes]
[, FORMAT RELEASE major.minor]
[, OBJECTDEFS | NO_OBJECTDEFS]
[, TRAILBYTEORDER {BIGENDIAN | LITTLEENDIAN | NATIVEENDIAN}]
```

file_name
The relative or fully qualified name of the file.

APPEND
Adds the current data to existing data in the file. If you use APPEND, do not use PURGE.

PURGE
Deletes an existing file before creating a new one. If you use PURGE, do not use APPEND.

MEGABYTES *megabytes*
Valid for Extract. The maximum size, in megabytes, of a file in the trail. The default is 2000.

FORMAT RELEASE *major.minor*
Specifies the metadata format of the data that is sent by Extract to the file. The metadata tells the reader process whether the data records are of a version that it supports. The metadata format depends on the version of the Oracle GoldenGate process. Older Oracle GoldenGate versions contain different metadata than newer ones.

- FORMAT is a required keyword.
- RELEASE specifies an Oracle GoldenGate release version. *major* is the major version number, and *minor* is the minor version number. The X.x must reflect a current or earlier, generally available (GA) release of Oracle GoldenGate. Valid

values are 9.0 through the current Oracle GoldenGate `X.x` version number, for example 11.2 or 12.1. (If you use an Oracle GoldenGate version that is earlier than 9.0, specify either 9.0 or 9.5.)

The release version is programmatically mapped back to an appropriate internal compatibility level. The default is the current version of the process that writes to this trail. Note that `RELEASE` versions earlier than 12.1 do not support three-part object names.

There is a dependency between `FORMAT` and the `RECOVERYOPTIONS` parameter. When `RECOVERYOPTIONS` is set to `APPENDMODE`, `FORMAT` must be set to `RELEASE 10.0` or greater. When `RECOVERYOPTIONS` is set to `OVERWRITEMODE`, `FORMAT` must be set to `RELEASE 9.5` or less.
The following settings are supported in Oracle Database 12.2:

- For Oracle Database 12.2 non-CDB with compatibility set to 12.1, FORMAT RELEASE 12.2 or above is supported.
- For Oracle Database 12.2 non-CDB with compatibility set to 12.2, FORMAT RELEASE 12.2 or above is supported.
- For Oracle Database 12.2 CDB/PDB with compatibility set to 12.2, only FORMAT RELEASE 12.3 is supported. This is due to the use of local undo for PDBs, which requires augmenting the transaction ID with the PDB number to ensure uniqueness of trx IDs.

`OBJECTDEFS | NO_OBJECTDEFS`
Use the `OBJECTDEFS` and `NO_OBJECTDEFS` options to control whether or not to include the object definitions in the trail. These two options are applicable only when the output trail is formatted in Oracle GoldenGate canonical format and the trail format release is greater than 12.1. Otherwise, both options are ignored because no metadata record will be added to the trail.
When replicating from an Open Systems database to NonStop, specify format version below 12.2 to avoid including the object definitions in the trail since NonStop does not support processing object definitions from the trail.

`TRAILBYTEORDER {BIGENDIAN | LITTLEENDIAN | NATIVEENDIAN}`
Sets the byte format of the metadata in the file records. This parameter does not affect the column data. Valid only for files that have a `FORMAT RELEASE` version of at least 12.1. Valid values are `BIGENDIAN` (big endian), `LITTLEENDIAN` (little endian), and `NATIVEENDIAN` (default of the local system). The default is `BIGENDIAN`. See the `GLOBALS` version of TRAILBYTEORDER for additional usage instructions.

Examples

Example 1

```
RMTFILE /ggs/dirdat/salesny, MEGABYTES 2, PURGE
```

Example 2

```
RMTFILE /ggs/dirdat/salesny, MEGABYTES 2, FORMAT RELEASE 10.4
```

2.132 RMTHOST

Valid For

Extract

Description

Use the RMTHOST parameter to:

- Identify a remote system to which the local Extract process connects
- Specify the TCP/IP port number on that system where the Manager process is running
- Control various attributes of the TCP/IP connections

This parameter controls compression, data encryption, buffer attributes, TCP/IP streaming, connection timeout threshold, and the wait period for a connection request. It also can be used to set Collector parameters.

To identify multiple remote systems in a parameter file, use one RMTHOST statement for each one, followed by the associated trails and table maps, for example:

```
EXTRACT sales
USERIDALIAS tiger1
RMTHOST ny, MGRPORT 7888, ENCRYPT AES192 KEYNAME mykey
RMTTRAIL /ggs/dirdat/aa
TABLE ora.orders;
RMTHOST la, MGRPORT 7888, ENCRYPT AES192 KEYNAME mykey2
RMTTRAIL /ggs/dirdat/bb
TABLE ora.orders;
```

Do not use RMTHOST for an Extract created in PASSIVE mode. .

Oracle GoldenGate supports IPv4 and IPv6 protocols. See USEIPV4 for more information about the selection of internet protocol.

The RMTHOST and RMTHOSTOPTIONS parameters can be specified together; the RMTHOST parameter is *not* required for RMTHOSTOPTIONS if the dynamic IP assignment is properly configured. When RMTHOSTOPTIONS is used, the MGRPORT option is ignored.

Default

None

Syntax

```
RMTHOST
{ host name | IP address}
[, COMPRESS]
[, COMPRESSTHRESHOLD]
[, ENCRYPT {BLOWFISH KEYNAME key_name | algorithm [KEYNAME key_name]} ]
{, MGRPORT port | PORT port}
[, PARAMS collector_parameters]
[, SOCKSPROXY {host_name | IP address} [:port] [PROXYCSALIAS credential_
store_alias [PROXYCSDOMAIN credential_store_domain]]
[, STREAMING | NOSTREAMING]
[, TCPBUFSIZE bytes]
[, TCPFLUSHBYTES bytes]
[, TIMEOUT seconds]
[, DGST SHA1|SHA2]
```

{host_name | IP_address}
The DNS host name or IP address of the target system. You can use either one to define the host. If using an IP address, use either an IPv6 or IPv4-mapped address, depending on the stack of the destination system.

COMPRESS
This option is valid for online or batch Extract processes and any Oracle GoldenGate initial-load method that uses trails. Compresses outgoing blocks of records to reduce bandwidth requirements. Oracle GoldenGate decompresses the data before writing it to the trail. `COMPRESS` typically results in compression ratios of at least 4:1 and sometimes better. However, compressing data can consume CPU resources.

COMPRESSTHRESHOLD
This option is valid for online or batch Extract processes and any Oracle GoldenGate initial-load method that uses trails. Sets the minimum block size for which compression is to occur. Valid values are from 0 and through 28000. The default is 1,000 bytes.

ENCRYPT *algorithm* [KEYNAME *key_name*]
This option is valid for online or batch Extract processes and all Oracle GoldenGate initial-load methods. Encrypts the data stream sent over TCP/IP to the target system. This option supports the following encryption options:

- **Master key and wallet method**: Generate a session key based on the active master key and algorithm specified. Not valid for BLOWFISH algorithm. Not valid for DB2 on z/OS, and DB2 for i.
- **ENCKEYS method**: Generate an AES encryption key, store it under a given name in an `ENCKEYS` file, and configure Oracle GoldenGate to use that key to encrypt the data.

  algorithm
  Specifies the encryption algorithm to use:

  - `AES128` uses the AES-128 cipher, which has a key size of 128 bits. AES128 is the default if no algorithm is specified.
  - `AES192` uses the AES-192 cipher, which has a key size of 192 bits.
  - `AES256` uses the AES-256 cipher, which has a key size of 256 bits.
  - `BLOWFISH` uses Blowfish encryption with a 64-bit block size and a variable-length key size from 32 bits to 128 bits. Use `BLOWFISH` for backward compatibility with earlier Oracle GoldenGate versions and for Oracle GoldenGate installations for DB2 z/OS and DB2 for i. On those platforms, `BLOWFISH` is the only supported encryption method. Use AES where supported, because it is more secure than `BLOWFISH` for those platforms.

  KEYNAME *key_name*
  Specifies that the `ENCKEYS` method of encryption will be used. Not valid for the master key and wallet method. For *key_name*, specify the logical name of the user-defined encryption key. Oracle GoldenGate uses the key name to look up the actual key in the `ENCKEYS` lookup file. To use the `ENCKEYS` method, you must:

  - Generate the encryption key.
  - Store it in an `ENCKEYS` lookup file.

- Copy ENCKEYS to every system where encryption or decryption (or both) are performed.

To use AES encryption for any database other than Oracle on a 32-bit platform, the path of the lib sub-directory of the Oracle GoldenGate installation directory must be specified as an environment variable before starting any processes. This is not required on 64-bit platforms. Set the path as follows:

- For Linux: Specify the path as an entry to the LD_LIBRARY_PATH. For example:

  ```
  setenv LD_LIBRARY_PATH ./lib:$LD_LIBRARY_PATH
  ```

- For Solaris: Specify the path as an entry to the SHLIB_PATH variable.
- For IBMi and AIX: Specify the path as an entry to the LIBPATH variable.
- For Windows: Add the path to the PATH variable.

You can use the SETENV parameter to set the library as a session variable for the process.
For more information about using encryption, see Overview of Security Options in *Securing the Oracle GoldenGate Environment*.

MGRPORT *port* | PORT *port*
Either MGRPORT or PORT is required. MGRPORT is the port on the remote system where Manager runs. PORT is the port number of a static Collector process. Either a Manager port (if using a dynamic Collector) or a static Collector port must be specified. See "Collector Parameters" for more information about a static Collector. The minimum is 1025 and the maximum is 65535.

SOCKSPROXY {*host_name* | *IP address*} *[:port]* [PROXYCSALIAS *credential_store_alias* [PROXYCSDOMAIN *credential_store_domain*]]
Use the SOCKSPROXY parameter to replicate information using a SOCKS5 proxy server creating a tunnel for TCP communication between a source Extract and a target process. The connection is initiated in the source side. You must specify the proxy address. Optionally you can specify the port or the default for SOCKS protocol will be used. If a credential store alias is specified, Oracle GoldenGate will use that information to authenticate with the proxy server. This is an option for RMTHOST parameter.
If there is no credential store information, no authentication with the proxy is performed.

host_name* | *IP_address
Use for an alias Extract. Specifies the DNS host name or IP address of the proxy server. You can use either one to define the host though you must use the IP address if your DNS server is unreachable. If you are using an IP address, use either an IPv6 or IPv4 mapped address, depending on the stack of the destination system.

port
(Optional) Specifies the port on the remote system where the proxy server accepts connections. The default value for port is 1080.

PROXYCSALIAS *credential_store_alias*
Specifies the credential store alias that resolves to the username and password used to authenticate with the proxy server.

PROXYCSDOMAIN *credential_store_domain*
(Optional) Specifies the credential store domain used together with the alias.

STREAMING | NOSTREAMING
This option is valid for online or batch Extract processes and any Oracle GoldenGate initial-load method that uses trails. Controls TCP/IP streaming.

STREAMING
Enables the asynchronous internet streaming protocol and is the default. In `STREAMING` mode, the receiver (Collector) does not send an acknowledgement to the sender (primary Extract or data pump) for any data packet unless the packet contains a flag requesting a response, typically when the sender must checkpoint or determine a write position. Because this method omits acknowledgements, the sender or receiver process terminates if there is a network disruption; therefore, when using `STREAMING`, use the `AUTORESTART` parameter in the Manager parameter file to restart Extract and Collector if they terminate.

NOSTREAMING
Enables the synchronous internet protocol. In `NOSTREAMING` mode, the sender sends a packet and then waits for the receiver to acknowledge it, before sending the next packet. This method is more reliable, because it enables the sender or receiver process to recover if there is a network disruption.

Extract falls back to the synchronous protocol automatically if the host system of the receiver process is not configured to use streaming.
Keep the `STREAMING` default unless you are requested to disable it, because streaming reduces transmission latency, especially in networks where latency is a problem already. Streaming is not supported for initial-load tasks where Extract communicates directly with Replicat.

TCPBUFSIZE *bytes*
This option is valid for online or batch Extract processes and any Oracle GoldenGate initial-load method that uses trails. Controls the size of the TCP socket buffer, in bytes, that Extract will try to maintain.
By increasing the size of the buffer, you can send larger packets to the target system.The actual size of the buffer depends on the TCP stack implementation and the network. The default is 30,000 bytes, but modern network configurations usually support higher values. Valid values are from 1000 to 200000000 (two hundred million) bytes. Work with your network administrator to determine an optimal value. See also *Administering Oracle GoldenGate* for more information about tuning the buffer size and other suggestions for improving the transfer of data across the network.
Testing has shown that using `TCPBUFSIZE` for initial loads produces three times faster throughput than loads performed without it. Do not use this parameter if the target system is NonStop.

TCPFLUSHBYTES *bytes*
This option is valid for online or batch Extract processes and any Oracle GoldenGate initial-load method that uses trails. Controls the size of the buffer, in bytes, that collects data that is ready to be sent across the network.
When either this value or the value of the `FLUSHSECS` parameter is reached, the data is flushed to the target. The default is 30,000 bytes. Valid values are from 1000 to 200000000 (two hundred million) bytes, but should be at least the value of `TCPBUFSIZE`.
Do not use this parameter for an initial load Extract. It is valid only for an online Extract group. Do not use this parameter if the target system is NonStop.

TIMEOUT *seconds*
This option is valid for online or batch Extract processes and any Oracle GoldenGate initial-load method that uses trails. Specifies how long Collector waits to get a connection from Extract, and how long Collector waits for a heartbeat signal from Extract before terminating a connection. Valid values are 1 second to 1800 seconds (30 minutes). The default value is 300 seconds (5 minutes). Setting the timeout to a very low value is not recommended in a production setting. You might need to increase the TIMEOUT value if you see a warning in the error log that there was a TCP/IP error 10054 (existing connection forcibly closed by remote host). This error typically occurs when the Collector terminates itself after the TIMEOUT value is exceeded. This parameter does not affect a static Collector.

DGST *SHA1|SHA2*
This is a new Extract option to specify the AES encryption method. This option is only valid when the RMTHOST parameter is used with PORT and ENCRYPT AES options using Oracle wallet. Valid value is either SHA1 or SHA2. SHA1 is default and works with the previous release of server collector. SHA2 only works with Server Collector 12.3. Both Extract and Server Collector must specify the same encryption method, otherwise the connection fails. Here's an example of using this option:
```
$ server -p 9050 -encrypt AES128 -dgst SHA2
```

Examples

Example 1

```
RMTHOST 20.20.20.17, MGRPORT 7809, ENCRYPT AES192, KEYNAME newyork
```

Example 2

```
RMTHOST 20.20.20.17, MGRPORT 7809, ENCRYPT AES192
```

Example 3

```
RMTHOST newyork, MGRPORT 7809, COMPRESS, COMPRESSTHRESHOLD 750, NOSTREAMING
```

Example 4

```
RMTHOST newyork, MGRPORT 7809, TCPBUFSIZE 100000, TCPFLUSHBYTES 300000
```

Example 5

```
RMTHOST newyork, MGRPORT 18819, CPU 1, PRI 140, HOMETERM $ZTN0.#PTJ52A1,
PROCESSNAME $xyz1
```

Example 6

```
RMTHOST lc01abc, MGRPORT 7809, SOCKSPROXY 192.111.82.180:3128 PROXYCSALIAS
proxyAlias PROXYCSDOMAIN support
```

2.133 RMTHOSTOPTIONS

Valid For

Passive Extract

Description

Use the RMTHOSTOPTIONS parameter to control attributes of a TCP/IP connection made between an Extract group running in PASSIVE mode on a less trusted source to a

target system in a more secure network zone. This parameter controls compression, data encryption, buffer attributes, streaming, and the wait period for a connection request. It also can be used to set Collector parameters.

This parameter differs from the RMTHOST parameter because it does not provide the host information needed to establish a remote connection. When Extract is running in PASSIVE mode, all connections between source and target are established by an alias Extract group on the target.

All parameter options must be specified in one RMTHOSTOPTIONS statement. If multiple RMTHOSTOPTIONS statements are used, the last one in the parameter file is used, and the others are ignored. RMTHOSTOPTIONS overrides any RMTHOST statements in the file.

See RMTHOST for additional information about supported IP protocols.

The RMTHOST and RMTHOSTOPTIONS parameters can be specified together; the RMTHOST parameter is *not* required for RMTHOSTOPTIONS if the dynamic IP assignment is properly configured. When RMTHOSTOPTIONS is used, the MGRPORT option is ignored.

Default

None

Syntax

```
RMTHOSTOPTIONS
[, COMPRESS]
[, COMPRESSTHRESHOLD]
[, ENCRYPT algorithm [KEYNAME key_name]]
[, PARAMS collector_parameters]
[, STREAMING | NOSTREAMING]
[, TCPBUFSIZE bytes]
[, TCPFLUSHBYTES bytes]
[, TIMEOUT seconds]
```

COMPRESS
Compresses outgoing blocks of records to reduce bandwidth requirements. Oracle GoldenGate decompresses the data before writing it to the trail. COMPRESS typically results in compression ratios of at least 4:1 and sometimes better. However, compressing data can consume CPU resources.

COMPRESSTHRESHOLD
Sets the minimum block size for which compression is to occur. Valid values are from 0 and through 28000. The default is 1,000 bytes.

ENCRYPT *algorithm* [KEYNAME *key_name*]
Encrypts the data stream sent over TCP/IP to the target system. This option supports the following encryption options:

- **Master key and wallet method**: Generate a one-time AES key to encrypt the data across the TCP/IP network. Then, the one-time key is encrypted by the master-key and stored in the trail file header.
- **ENCKEYS method**: Generate an AES encryption key, store it under a given name in an ENCKEYS file, and configure Oracle GoldenGate to use that key to directly encrypt the data across the TCP/IP network.

algorithm
Specifies the encryption algorithm to use:

- `AES128` uses the AES-128 cipher, which has a key size of 128 bits. AES128 is the default if no algorithm is specified.
- `AES192` uses the AES-192 cipher, which has a key size of 192 bits.
- `AES256` uses the AES-256 cipher, which has a key size of 256 bits.
- `BLOWFISH` uses Blowfish encryption with a 64-bit block size and a variable-length key size from 32 bits to 128 bits. Use `BLOWFISH` for backward compatibility with earlier Oracle GoldenGate versions and for Oracle GoldenGate installations for DB2 on z/OS, DB2 for i, and SQL/MX on NonStop. On those platforms, `BLOWFISH` is the only supported encryption method. Use AES where supported, because it is more secure than `BLOWFISH` for those platforms.

KEYNAME *key_name*
Specifies that the `ENCKEYS` method of encryption will be used. Not valid for the master key and wallet method. For *key_name*, specify the logical name of the user-defined encryption key. Oracle GoldenGate uses the key name to look up the actual key in the `ENCKEYS` lookup file. To use the `ENCKEYS` method, you must:

- Generate the encryption key.
- Store it in an `ENCKEYS` lookup file.
- Copy `ENCKEYS` to every system where encryption or decryption (or both) are performed.

To use AES encryption for any database other than Oracle on a 32-bit platform, the path of the `lib` sub-directory of the Oracle GoldenGate installation directory must be specified as an environment variable before starting any processes. This is not required on 64-bit platforms. Set the path as follows:

- Linux: Specify the path as an entry to the `LD_LIBRARY_PATH`. For example:

  ```
  setenv LD_LIBRARY_PATH ./lib:$LD_LIBRARY_PATH
  ```

- For Solaris: Specify the path as an entry to the `SHLIB_PATH` variable.
- For IBMi and AIX: Specify the path as an entry to the `LIBPATH` variable.
- Windows: Add the path to the `PATH` variable.

You can use the `SETENV` parameter to set the library as a session variable for the process.

PARAMS *collector_parameters*
Specifies Collector parameters on a NonStop target system.

> **Note:**
>
> Do not specify a Collector port (`-p` argument) if Manager will be starting Collector dynamically.

For more information about Collector parameters on the NonStop platform, see *Reference Guide for Oracle GoldenGate for HP NonStop (Guardian)*.

`STREAMING | NOSTREAMING`
Controls TCP/IP streaming.

`STREAMING`
Enables the asynchronous internet streaming protocol and is the default. In `STREAMING` mode, the receiver (Collector) does not send an acknowledgement to the sender (primary Extract or data pump) for any data packet unless the packet contains a flag requesting a response, typically when the sender must checkpoint or determine a write position. Because this method omits acknowledgements, the sender or receiver process terminates if there is a network disruption; therefore, when using `STREAMING`, use the `AUTORESTART` parameter in the Manager parameter file to restart Extract and Collector if they terminate.

`NOSTREAMING`
Enables the synchronous internet protocol. In `NOSTREAMING` mode, the sender sends a packet and then waits for the receiver to acknowledge it, before sending the next packet. This method is more reliable, because it enables the sender or receiver process to recover if there is a network disruption.

Extract falls back to the synchronous protocol automatically if the host system of the receiver process is not configured to use streaming.
Keep the `STREAMING` default unless you are requested to disable it, because streaming reduces transmission latency, especially in networks where latency is a problem already. Streaming is not supported for initial-load tasks where Extract communicates directly with Replicat.

`TCPFLUSHBYTES` *`bytes`*
Controls the size of the buffer, in bytes, that collects data that is ready to be sent across the network. When either this value or the value of the `FLUSHSECS` parameter is reached, the data is flushed to the target.
The default is 30,000 bytes. Valid values are from 1000 to 200000000 (two hundred million) bytes, but should be at least the value of `TCPBUFSIZE`.
Do not use this parameter for an initial load Extract. It is valid only for an online Extract group. Do not use this parameter if the target system is NonStop.

`TIMEOUT` *`seconds`*
Specifies how long an Extract running in `PASSIVE` mode waits to get a connection from Collector, and how long Extract waits for a heartbeat signal from Collector before terminating a connection. Valid values are 1 second to 1800 seconds (30 minutes). The default value is 300 seconds (5 minutes). Setting the timeout to a very low value is not recommended in a production setting. You might need to increase the `TIMEOUT` value if you see a warning in the error log that there was a TCP/IP error 10054 (existing connection forcibly closed by remote host). This error typically occurs when the Extract terminates itself after the `TIMEOUT` value is exceeded.

Example

```
RMTHOSTOPTIONS ENCRYPT AES192, KEYNAME newyork, COMPRESS, COMPRESSTHRESHOLD 750,
TCPBUFSIZE 100000, TCPFLUSHBYTES 300000, NOSTREAMING
```

2.134 RMTTASK

Valid For

Extract

Description

Use the RMTTASK parameter for an initial-load Extract to initiate a Replicat processing task during an Oracle GoldenGate direct load or a direct bulk load to SQL*Loader. RMTTASK directs Extract to communicate directly with Replicat over TCP/IP and bypasses the use of a Collector process or trail storage. RMTTASK also directs Extract to request that Manager start Replicat automatically, and then stop Replicat when the run is finished. Tasks do not use checkpoints.

Dependent parameters are as follows:

- A RMTHOST statement must follow each RMTTASK statement in the initial-load Extract parameter file.
- EXTRACT must be used in the initial-load Extract parameter file.
- REPLICAT must be used in the initial-load Replicat parameter file.
- SOURCEISTABLE must be used in the ADD EXTRACT command.
- SPECIALRUN must be used in the ADD REPLICAT command.

RMTTASK does not support encryption of any kind. To use encryption, you can use the initial-load method that writes data to a file, which is read by Replicat to load the data.

RMTTASK supports all Oracle data types, including BLOB, CLOB, NCLOB, LONG, UDT, and XML.

When using RMTTASK, do not start Replicat with the START REPLICAT command. Replicat is started automatically during the task.

See the *Administering Oracle GoldenGate* for more information about performing initial data loads.

Default

None

Syntax

```
RMTTASK REPLICAT, GROUP group_name
[FORMAT RELEASE major.minor]
```

GROUP *group_name*
The group name of the Initial Load Replicat on the target system.

FORMAT RELEASE *major.minor*
Specifies the metadata format of the data that is sent by Extract to Replicat. The metadata tells Replicat whether the data records are of a version that it supports. The metadata format depends on the version of the Oracle GoldenGate process. Older Oracle GoldenGate versions contain different metadata than newer ones.

- FORMAT is a required keyword.
- RELEASE specifies an Oracle GoldenGate release version. *major* is the major version number, and *minor* is the minor version number. The X.x must reflect a current or earlier, generally available (GA) release of Oracle GoldenGate. Valid values are 9.0 through the current Oracle GoldenGate X.x version number, for example 11.2 or 12.1. (If you use an Oracle GoldenGate version that is earlier than 9.0, specify either 9.0 or 9.5.)

  The release version is programmatically mapped back to an appropriate internal compatibility level. The default is the current version of the process that writes to this trail. Note that RELEASE versions earlier than 12.1 do not support three-part object names.

There is a dependency between FORMAT and the RECOVERYOPTIONS parameter. When RECOVERYOPTIONS is set to APPENDMODE, FORMAT must be set to RELEASE 10.0 or greater. When RECOVERYOPTIONS is set to OVERWRITEMODE, FORMAT must be set to RELEASE 9.5 or less.
See *Administering Oracle GoldenGate* for more information about initial loads.

Example

```
RMTTASK REPLICAT, GROUP initrep, FORMAT RELEASE 10.0
```

2.135 RMTTRAIL

Valid For

Extract

Description

Use the RMTTRAIL parameter to specify a remote trail that was created with the ADD RMTTRAIL command in GGSCI. A trail specified with RMTTRAIL must precede its associated TABLE statements. Multiple RMTTRAIL statements can be used to specify different remote trails. RMTTRAIL must be preceded by a RMTHOST parameter.

RMTTRAIL parameter is deprecated and ignored for Data Pump

You can encrypt the data in this trail by using the ENCRYPTTRAIL parameter. See "ENCRYPTTRAIL | NOENCRYPTTRAIL" for more information.

Default

None

Syntax

```
RMTTRAIL trail_name
[, FORMAT RELEASE major.minor]
[, OBJECTDEFS | NO_OBJECTDEFS]
[, TRAILBYTEORDER {BIGENDIAN | LITTLEENDIAN | NATIVEENDIAN}]
```

name
The relative or fully qualified path name of the trail. Use two characters for the name. As trail files are aged, a six-character sequence number will be added to this name, for example /ggs/dirdat/rt000000001.

RELEASE specifies an Oracle GoldenGate release version. *major* is the major version number, and *minor* is the minor version number. The X.x must reflect a current or earlier, generally available (GA) release of Oracle GoldenGate. Valid values are 9.0 through the current Oracle GoldenGate X.x version number. For the Oracle GoldenGate 18c (18.1.0.0) release, the valid value is 18.1. If you use an Oracle GoldenGate version that is earlier than 9.0, specify either 9.0 or 9.5.

RELEASE specifies an Oracle GoldenGate release version. *major* is the major version number, and *minor* is the minor version number. The X.x must reflect a current or earlier, generally available (GA) release of Oracle GoldenGate. Valid values are 9.0 through the current Oracle GoldenGate X.x version number. For the Oracle GoldenGate 18c (18.1.0.0) release, the valid value is 18.1. If you use an Oracle GoldenGate version that is earlier than 9.0, specify either 9.0 or 9.5.

FORMAT RELEASE *major.minor*
Specifies the metadata format of the data that is sent by Extract to the trail. The metadata tells the reader process whether the data records are of a version that it supports. Older Oracle GoldenGate versions contain different metadata than newer ones.

- FORMAT is a required keyword.
- RELEASE specifies an Oracle GoldenGate release version. *major* is the major version number, and *minor* is the minor version number. The X.x must reflect a current or earlier, generally available (GA) release of Oracle GoldenGate. Valid values are 9.0 through the current Oracle GoldenGate X.x version number, for example 11.2 or 12.1. (If you use an Oracle GoldenGate version that is earlier than 9.0, specify either 9.0 or 9.5.)

  RELEASE specifies an Oracle GoldenGate release version. major is the major version number, and minor is the minor version number. The X.x must reflect a current or earlier, generally available (GA) release of Oracle GoldenGate. Valid values are 9.0 through the current Oracle GoldenGate X.x version number. For the Oracle GoldenGate 18c (18.1.0.0) release, the valid value is 18.1. If you use an Oracle GoldenGate version that is earlier than 9.0, specify either 9.0 or 9.5.

  The release version is programmatically mapped back to an appropriate internal compatibility level. The default is the current version of the process that writes to this trail. Note that RELEASE versions earlier than 12.1 do not support three-part object names.

There is a dependency between FORMAT and the RECOVERYOPTIONS parameter. When RECOVERYOPTIONS is set to APPENDMODE, FORMAT must be set to RELEASE 10.0 or greater. When RECOVERYOPTIONS is set to OVERWRITEMODE, FORMAT must be set to RELEASE 9.5 or less.
See *Administering Oracle GoldenGate* for additional information about Oracle GoldenGate trail file versioning and recovery modes.
The following settings are supported in Oracle Database 12.2:

- For Oracle Database 12.2 non-CDB with compatibility set to 12.1, FORMAT RELEASE 12.2 or above is supported.
- For Oracle Database 12.2 non-CDB with compatibility set to 12.2, FORMAT RELEASE 12.2 or above is supported.

- For Oracle Database 12.2 CDB/PDB with compatibility set to 12.2, only the FORMAT RELEASE values 12.3 and 18.1 are supported. This is due to the use of local undo for PDBs, which requires augmenting the transaction ID with the PDB number to ensure uniqueness of trx IDs.

OBJECTDEFS | NO_OBJECTDEFS
Use the OBJECTDEFS and NO_OBJECTDEFS options to control whether or not to include the object definitions in the trail. These two options are applicable only when the output trail is formatted in Oracle GoldenGate canonical format and the trail format release is greater than 12.1. Otherwise, both options are ignored because no metadata record will be added to the trail.
When replicating from an Open Systems database to NonStop, specify format version below 12.2 to avoid including the object definitions in the trail since NonStop does not support processing object definitions from the trail.

TRAILBYTEORDER {BIGENDIAN | LITTLEENDIAN | NATIVEENDIAN}
Sets the byte format of the metadata in the trail records. This parameter does not affect the column data. Valid only for trails that have a FORMAT RELEASE version of at least 12.1. Valid values are BIGENDIAN (big endian), LITTLEENDIAN (little endian), and NATIVEENDIAN (default of the local system). The default is BIGENDIAN. See the GLOBALS version of TRAILBYTEORDER for additional usage instructions.

Examples

Example 1

```
RMTTRAIL dirdat/ny
```

Example 2

```
RMTTRAIL /ggs/dirdat/ny, FORMAT RELEASE 10.4
```

Example 3
Two trail formats within the same sets of tables being captured:

```
FORMAT RELEASE  11.2
TABLE tab1
TABLE tab2
FORMAT RELEASE 12.1
TABLE tab1
TABLE tab2
```

Example 4
Example of a data pump parameter file that sends an HR schema with object definitions and an ORD schema without object definitions:

```
RMTTRAIL $data/ggs12.2/a1, OBJECTDEFS
TABLE hr.*;
RMTTRAIL $data/ggs12.2/a2, NO_OBJECTDEFS
TABLE ord.*;
```

2.136 ROLLOVER

Valid For

Extract

Description

Use the ROLLOVER parameter to specify the interval at which trail files are aged and new ones are created. ROLLOVER is global and applies to all trails defined with RMTTRAIL or RMTFILE statements in a parameter file.

Use ROLLOVER to create trail files that represent distinct periods of time (for example, each day). It facilitates continuous processing while providing a means for organizing the output. It also provides a means for organizing batch runs by deactivating one file and starting another for the next run.

Files roll over between transactions, not in the middle of one, ensuring data integrity. Checkpoints are recorded when files roll over to ensure that previous files are no longer required for processing.

Rollover occurs only if the rollover conditions are satisfied during the run. For example, if ROLLOVER ON TUESDAY is specified, and data extraction starts on Tuesday, the rollover does not occur until the next Tuesday (unless more precise ROLLOVER rules are specified). You can specify up to 30 rollover rules.

Either the AT or ON option is required. Both options can be used together, and in any order. Using AT without ON creates a new trail file at the specified time every day.

A trail sequence number can be incremented from 000001 through 999999, and then the sequence numbering starts over at 000000.

Default

Roll over when the default file size is reached or the size specified with the MEGABYTES option of the ADD RMTTRAIL or ADD EXTTRAIL command is reached.

Syntax

```
ROLLOVER {AT hh:mi | ON day | AT hh:mi ON day} [REPORT]
```

AT *hh:mi*
The time of day to age the file.
Valid values:

- hh is based on a 24-hour clock, with valid values of 1 through 23.
- mi accepts values from 00 through 59.

ON *day*
The day of the week to age the file.
Valid values:

```
SUNDAY
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
```

The values are not case-sensitive.

REPORT
Generates a report for the number of records extracted from each table since the last report was generated. The report represents the number of records output to the corresponding trail unless other reports are generated by means of the REPORT parameter.

Examples

Example 1
The following ages trails every day at 3:00 p.m.

```
ROLLOVER AT 15:00
```

Example 2
The following ages trails every Sunday at 8:00 a.m.

```
ROLLOVER AT 08:00 ON SUNDAY
```

2.137 SCHEMAEXCLUDE

Valid For

Extract, Replicat, DEFGEN

Description

Use the SCHEMAEXCLUDE parameter to exclude source objects that are owned by the specified source owner (such as a schema) from the Oracle GoldenGate configuration when wildcards are being used to specify the owners in TABLE or MAP statements. This parameter is valid for two- and three-part names.

Wildcards can be used for the optional *catalog* or *container* specification, as well as the *schema* specification. Make certain not to use wildcards such that all objects are excluded. Follow the rules for using wildcards in *Administering Oracle GoldenGate*.

The positioning of SCHEMAEXCLUDE in relation to parameters that specify files or trails determines its effect. Parameters that specify trails or files are: EXTFILE, RMTFILE, EXTTRAIL, RMTTRAIL. The parameter works as follows:

- When a SCHEMAEXCLUDE specification is placed before any TABLE or SEQUENCE parameters, and also before the parameters that specify trails or files, it applies globally to all trails or files, and to all TABLE and SEQUENCE parameters.
- When a SCHEMAEXCLUDE specification is placed after a parameter that specifies a trail or file, it is effective only for that trail or file and only for the TABLE or SEQUENCE parameters that are associated with it. Multiple trail or file specifications can be made in a parameter file, each followed by a set of TABLE, SEQUENCE, and TABLEEXCLUDE specifications.

SCHEMAEXCLUDE is evaluated before evaluating the associated TABLE or SEQUENCE parameters. Thus, the order in which they appear does not make a difference.

See also the EXCLUDEWILDCARDOBJECTSONLY parameter.

Default

None

Syntax

```
SCHEMAEXCLUDE [container.]schema
```

***container*.**
If the database requires three-part names, specifies the source Oracle container that contains the source owner that is to be excluded. Use if a qualifier is required to identify the correct owner to exclude.

schema
Specifies the name of the source owner that is to be excluded. For databases that require three-part names, you can use *schema* without *container* if the SCHEMAEXCLUDE specification precedes a set of TABLE or MAP parameters for which the default container is specified with the SOURCECATALOG parameter.

Examples

Example 1
This Oracle example requires both *container* and *schema* specifications and demonstrates how wildcards can be used as part of the specification.

```
EXTRACT capt
USERIDALIAS alias1
RMTHOST sysb, MGRPORT 7809
RMTTRAIL /ggs/dirdat/aa
SCHEMAEXCLUDE pdbtest.test*
TABLE pdb*.*.*;
```

Example 2
This example shows how to use SCHEMAEXCLUDE when the database requires only a two-part name.

```
TABLE abc*.*;
SCHEMAEXCLUDE abctest*
```

2.138 SEQUENCE

Valid For

Extract

Description

Use the SEQUENCE parameter to capture sequence values from the transaction log. Currently, Oracle GoldenGate supports sequences for the Oracle database.

> **Note:**
>
> DDL support for sequences (CREATE, ALTER, DROP, RENAME) is compatible with, but not required for, replicating sequence values. To replicate just sequence values, you *do not* need to install the Oracle GoldenGate DDL support environment. You can just use the SEQUENCE parameter.

Oracle GoldenGate ensures that the values of a target sequence are:

- higher than the source values if the increment interval is positive
- lower than the source values if the increment interval is negative

Depending on the increment direction, Replicat applies one of the following formulas as a test when it performs an insert:

source_highwater_value + (*source_cache_size* * *source_increment_size*) = *target_highwater_value*

Or...

source_highwater_value + (*source_cache_size* * *source_increment_size*) >= *target_highwater_value*

If the formula evaluates to FALSE, the target sequence is updated to be higher than the source value (if sequences are incremented) or lower than the source value (if sequences are decremented). The target must always be ahead of, or equal to, the expression in the parentheses in the formula. For example, if the source high water value is 40, and CACHE is 20, the target high water value should be at least 60:

```
40 + (20*1) <60
```

If the target high water value is less than 80, Oracle GoldenGate updates the sequence to increase the high water value, so that the target remains ahead of the source. To get the current high water value, perform this query:

```
SELECT last_number FROM all_sequences WHERE
sequence_owner=upper('SEQUENCEOWNER') AND sequence_name=upper('SEQUENCENAME');
```

Supported Processing Modes

The processing modes that support the capture of sequences are as follows:

- Oracle GoldenGate supports sequences in an active-passive high-availability configuration. Oracle GoldenGate does not support the replication of sequence values in an active-active configuration. An active-passive configuration includes a primary Extract, a data pump, and a Replicat on both servers, but the processes are active in only one direction. The Extract process on the failover server must be inactive, which includes not capturing sequences. See the *Administering Oracle GoldenGate* for more information about how to configure Oracle GoldenGate for high-availability.
- If using SEQUENCE for a primary Extract that writes to a data pump, you must also use an identical SEQUENCE parameter in the data pump.
- Oracle GoldenGate initial load methods that contain the SOURCEISTABLE parameter, either as an Extract parameter or within ADD EXTRACT, do not support the replication of sequence values.

Guidelines for Using SEQUENCE

- The cache size and the increment interval of the source and target sequences must be identical.
- The cache can be any size, including 0 (NOCACHE).
- The sequence can be set to cycle or not cycle, but the source and target databases must be set the same way.

- To add SEQUENCE to a configuration in which DDL support is enabled, you must re-install the Oracle GoldenGate DDL objects in INITIALSETUP mode.

Error Handling

- If Extract cannot resolve a sequence name, it ignores the operation.
- To enable Replicat error handling for sequences, use the REPERROR parameter. This parameter is available as an option in the MAP parameter and also as a standalone parameter. REPERROR can detect if a sequence has been dropped on the target and can be used to retry a sequence operation until the sequence is recreated.
- REPERROR does not handle missing objects on startup. Use DDLERROR with IGNOREMISSINGTABLES.

Other Important Information

- Gaps are possible in the values of the sequences that Oracle GoldenGate replicates because gaps are inherent, and expected, in the way that sequences are maintained by the database. However, the target values will always be greater than those of the source.
- If Extract is running in single-threaded mode on a RAC system, and if sequences are updated on a node that has lag, it might take more time to capture a sequence. This is normal behavior.
- In a failover, any problem that causes the loss or corruption of data in a transaction log or Oracle GoldenGate trail file will cause the loss of the replicated sequence updates.
- The statistics shown by SEND EXTRACT and SEND REPLICAT when used with the REPORT option will show the sequence operation as an UPDATE.

Default

None

Syntax

```
SEQUENCE [container.]schema.sequence;
```

[*container*.]*schema.sequence*
Specifies the fully qualified name of the source sequence. Include the name of the pluggable database if the source is an Oracle container database. To specify object names and wildcards correctly, see *Administering Oracle GoldenGate*.

;
Terminates the SEQUENCE parameter statement.

Example

```
SEQUENCE hr.employees_seq;
```

2.139 SESSIONCHARSET

Valid For

GLOBALS, valid for MySQL

Description

Use the SESSIONCHARSET parameter to set the database session character set for all database connections that are initiated by Oracle GoldenGate processes in the local Oracle GoldenGate instance. Processes that log into the database include GGSCI, DEFGEN, Extract, and Replicat.

This parameter supports MySQL. The database character set for other databases is obtained programmatically.

The SESSIONCHARSET option of the DBLOGIN command can be used to override this setting for any commands issued in the same GGSCI session. The SESSIONCHARSET option of the SOURCEDB and TARGETDB parameters can be used to override this setting for individual process logins.

Default

Character set of the operating system

Syntax

```
SESSIONCHARSET character_set
```

character_set
The database session character set.

Example

```
SESSIONCHARSET ISO-8859-11
```

2.140 SETENV

Valid For

Extract and Replicat

Description

Use the SETENV parameter to set a value for an environment variable. When Extract or Replicat starts, it uses the specified value instead of the one that is set in the operating system environment. A variable set in the SETENV statement overrides any existing variables set at the operating system level. Use one SETENV statement per variable to be set.

For integrated extracts, you can set new environment variables if they are available from the lcr server. The new environment variables are:

- USERNAME: Database login user name
- OSUSERNAME: Operating System user name
- MACHINENAME: Name of the host, machine, or server where the database is running
- PROGRAMNAME: Name of program or application that started the transaction or session
- CLIENTIDENTIFIER: Value set using DBMS_SESSION.set_identifier()

SETENV cannot be used with query parameters.

Default

None

Syntax

```
SETENV (
{environment_variable |
  GGS_CacheRetryCount |
  GGS_CacheRetryDelay}
= 'value'
)
```

environment_variable
The name of the environment variable to be set.

'value'
A value for the specified variable. Enclose the value within single quotes.

GGS_CacheRetryCount
(SQL Server) Oracle GoldenGate environment parameter that controls the number of times that Extract tries to read the source transaction log files when they are blocked because of excessive system activity. The default is 10 retries. After trying the specified number of times, Extract abends with an error that begins as follows:

```
GGS ERROR 600 [CFileInfo::Read] Timeout expired after 10 retries with 1000 ms
delay waiting to read transaction log or backup files.
```

If you continue to see timeout messages in the report file or error log, increase this parameter to allow more retries.

GGS_CacheRetryDelay
(SQL Server) Oracle GoldenGate environment parameter that controls the number of milliseconds that Extract waits before trying again to read the transaction logs when the previous attempt has failed. The default is 1000 milliseconds delay.

Examples

Example 1
Using separate `SETENV` statements allows a single instance of Oracle GoldenGate to connect to multiple Oracle database instances without having to change environment settings. The following parameter statements set a value for `ORACLE_HOME` and `ORACLE_SID`.

```
SETENV (ORACLE_HOME = '/home/oracle/ora9/product')
SETENV (ORACLE_SID = 'ora9')
```

Example 2
The following parameter statements set values for Oracle GoldenGate in a SQL Server environment where Extract tries to read the transaction log for a maximum of 20 times before abending, with a delay of 3000 milliseconds between tries.

```
SETENV (GGS_CacheRetryCount = 20)
SETENV (GGS_CacheRetryDelay = 3000)
```

2.141 SHOWSYNTAX

Valid For

Replicat

Description

Use the SHOWSYNTAX parameter to start an interactive session where you can view each Replicat SQL statement before it is applied. Viewing SQL statements that failed may help you diagnose the cause of the problem. For example, you could find out that the WHERE clause is using a non-indexed column.

As long as a data type can be applied with dynamic SQL and the column data is bound with a SQL statement, Replicat shows some or all of the data in string form, hexadecimal form, or as a data identifier, depending on the data type. By default, Replicat does not show LOB data or other data types that are treated as a LOB by the database or by Oracle GoldenGate, whether or not the data is bound in SQL. Examples are LOB, XML, and UDT data types. Instead, Replicat shows a data identifier, for example "<LOB data>." To display this type of data, specify the INCLUDELOB option of SHOWSYNTAX. If the column data is not bound in a SQL statement, Replicat does not show the data even when INCLUDELOB is used.

If CHAR/VARCHAR/CLOB or NCHAR/NVARCHAR/NCLOB character data has an unprintable character (U+0000 to U+001F), the character is escaped and displayed in \xx form, where *xx* is a decimal value that ranges from 00 to 31.

The first time that you use SHOWSYNTAX, request guidance from Oracle Support. It is a debugging parameter and can cause unwanted results if used improperly. It requires manual intervention, which suspends automated processing and can cause backlogs and latency. Use SHOWSYNTAX in a test environment. Create duplicates of your Replicat groups and target tables so that the production environment is not affected.

SHOWSYNTAX is not supported for a coordinated Replicat group.

If used for an integrated Replicat group, sqltrace is enabled for the associated database apply process.

If you capture XML column data using Integrated Extract, the column is captured as updated column even if you do not update the column. As a result of this behavior, SHOWSYNTAX shows the XML column as updated column. However, if you capture the table using Classic Extract, the XML column does not appear in the SHOWSYNTAX SQL statement if the column is not part of the update.

To use SHOWSYNTAX, Replicat must be started from the command shell of the operating system. Do not use SHOWSYNTAX if Replicat is started through GGSCI.

BATCHSQL processing is suspended when SHOWSYNTAX is running. BATCHSQL mode is resumed when Replicat is re-started without SHOWSYNTAX.

To use SHOWSYNTAX, do the following:

1. From the Oracle GoldenGate home directory, start Replicat from the command shell of the operating system using the following syntax. This syntax eliminates the reportfile option and directs the output to the screen.

   ```
   replicat paramfile dirprm/Replicat_name.prm
   ```

2. The first SQL statement is displayed with some prompts.
   - Choose `Keep Displaying` (the default) to execute the current statement and display the next one.
   - Choose `Stop Display` to resume normal processing and stop printing SQL statements to screen.
3. When finished viewing syntax, remove `SHOWSYNTAX` from the parameter file.

Default

None

Syntax

```
SHOWSYNTAX [APPLY | NOAPPLY] [INCLUDELOB [max_bytes | ALL]]
```

`APPLY | NOAPPLY`
Controls whether or not Replicat applies the data that is displayed with `SHOWSYNTAX` to the target database. The default is `APPLY` (apply the data to the target database). `NOAPPLY` prevents the application of the data to the target and does not write the records to the discard file.

`INCLUDELOB [max_bytes] | ALL`
Includes `LOB`, `XML`, and `UDT` data in the `SHOWSYNTAX` output. Without this option, only a data identifier is displayed, such as "`<LOB data>`." The default is 2.

`max_bytes`
Specifies the maximum length of `LOB`, `XML`, or `UDT` data that is displayed. Valid units are `K`, `M`, or `G`. The default is to display the first 2K bytes.

`ALL`
Displays `LOB` data in its entirety.

Example

```
SHOWSYNTAX INCLUDELOB 1M
```

2.142 SOURCEDB

Valid For

Manager, Extract, DEFGEN

Description

Use the `SOURCEDB` parameter for databases or data sets that require a data source name or identifier to be specified explicitly as part of the connection information. This option is required to identify one of the following:

- The source database for heterogeneous databases.
- The data source name (DSN) for supported databases that use ODBC
- The default DB2 for i database.

Tables specified in `TABLE` statements that follow `SOURCEDB` are assumed to be from the specified data source.

You might need to use the USERID or USERIDALIAS parameter in the SOURCEDB parameter statement, depending on the authentication that is required for the data source.

For databases that allow authentication at the operating-system level, you can specify SOURCEDB without USERID or USERIDALIAS.

For DB2 LUW, the SOURCEDB statement must refer to the database by its real name, rather than by any alias.

See USERID | NOUSERID or USERIDALIAS for more information.

Default

None

Syntax

```
SOURCEDB data_source[, SESSIONCHARSET character_set]
```

data_source
The name of the database, catalog, or data source name as applicable for the database.
For MySQL databases, you can use the format of SOURCEDB *database_name@host_name* to avoid connection issues caused by the incorrect configuration of localhost in the local hosts file. If running MySQL on a port other than the default of 3306, you must specify the port number in the connect string: SOURCEDB *database_name@host_name*:*port*.

SESSIONCHARSET *character_set*
Supports MySQL. Sets the database session character set for the process login session. This parameter overrides any SESSIONCHARSET that is specified in the GLOBALS file.

Examples

Example 1
This example shows SOURCEDB using OS-level authentication.

```
SOURCEDB mydb
```

Example 2
This example shows SOURCEDB with the USERIDALIAS parameter.

```
SOURCEDB mydb, USERIDALIAS tiger1
```

2.143 SOURCECATALOG

Valid For

Extract and Replicat

Description

Use the SOURCECATALOG parameter to specify one of the following for subsequent TABLE or MAP statements that contain two-part names, where three-part object names are required to fully identify a default source Oracle pluggable database (PDB)

This parameter provides an efficient alternative to specifying the full three-part object name (`container.schema.object` or `catalog.schema.object`) when specifying source objects from an Oracle consolidated database. Only the two-part name (`schema.object`) need be specified in subsequent `TABLE` or `MAP` statements when `SOURCECATALOG` is used. You can use multiple instances of `SOURCECATALOG` to specify different default containers or catalogs for different sets of `TABLE` statements (or `SEQUENCE` statements, if Oracle).

Three-part name specifications encountered after `SOURCECATALOG` override the `SOURCECATALOG` specification in a `TABLE` statement, `MAP` statement, or other parameter that takes object names as input.

Default

None

Syntax

```
SOURCECATALOG {container}
```

container
The name of an Oracle pluggable database that contains the specified objects in the `TABLE` of `MAP` statement.

Example

In the following example, `SOURCECATALOG` is used to specify three different source Oracle PDBs in an Extract parameter file.

```
SOURCECATALOG FINANCE
TABLE SAP.*;
TABLE REPORTS.*;
SOURCECATALOG HR
TABLE SIEBEL.*;
TABLE REPORTS.*;
SOURCECATALOG MFG
TABLE CUSTOMER.ORDERS;
TABLE REPORTS.*;
TABLE HQ.LOCATIONS.*;
```

In this example, Extract captures the following:

- All tables in the `SAP` and `REPORTS` schemas in the `FINANCE` PDB.
- All tables in the `SIEBEL` and `REPORTS` schemas in the `HR` PDB.
- All tables in the `CUSTOMER` and `REPORTS` schemas in the `MFG` PDB.
- For the last `TABLE` statement, Extract captures all tables in the `LOCATIONS` schema in the `HQ` PDB. This statement is a fully qualified three-part name and overrides the previous `SOURCECATALOG` specification.

2.144 SOURCECHARSET

Valid For

Replicat

Description

Use the SOURCECHARSET parameter to control the conversion of data from the source character set to the target character set by Replicat. Replicat converts character sets by default for versions 11.2.1 and later, but you may need to intervene in the following cases:

- To enable accurate conversion of data written by an Extract version earlier than 11.2.1. Extract versions prior to version 11.2.1 do not write information about the source character set to the trail, so the information must be supplied to Replicat directly. Extract versions 11.2.1 and later write information about the source character set to the trail for use by Replicat, and any SOURCECHARSET specification is ignored.
- To override the source database character set in the trail file. Use SOURCECHARSET with the OVERRIDE option to specify the character set you want to use. An example use case is migrating a database to UNICODE or particular character set database from garbage in, garbage out type of non-character set aware database by ignoring the source database character set.

Replicat issues a warning message when it uses the SOURCECHARSET character set.

Use the REPLACEBADCHAR parameter to handle validation errors where there are invalid characters in the source data or the target character set does not support a source character. It provides options to abend on these errors, skip the record that caused the error, or specify a substitute value for the character.

Default

None

Syntax

```
SOURCECHARSET {source_charset | PASSTHRU | OVERRIDE} [DB2ZOS]
```

source_charset
Specifies the source character set for data that is written by an Extract version that is earlier than 11.2.1. Replicat uses the specified character set when converting character-type columns to the target character set.
For *source_charset*, specify the appropriate character-set identifier that represents the source database. For a list of supported character sets, see *Administering Oracle GoldenGate*.
For Oracle, if SOURCECHARSET is not specified but there is an NLS_LANG environment variable on the target, Replicat uses the NLS_LANG value as the source database character set. If neither SOURCECHARSET nor NLS_LANG is present, Replicat abends to prevent possible data corruption.

PASSTHRU
PASSTHRU
Forces Replicat to apply the data without converting the character set. Character set differences are ignored as follows:

- If the database is Oracle, the data is applied the way it is stored in the trail.
- If the database is other than Oracle, the data is applied as binary data if the database supports a bind as binary data. Otherwise, the data is applied as-is.

PASSTHRU is not compatible with the BULKLOAD parameter (direct-bulk load).
If PASSTHRU is specified and a mapping between CHAR/VARCHAR/CLOB and NCHAR/NVARCHAR/NCLOB exists in the MAP statement, Replicat abends.
If any Oracle GoldenGate column-mapping functions are used for character-based columns when PASSTHRU mode is specified, Replicat issues a warning message and converts the results of those functions to the target database character set before mapping them to the target column.
PASSTHRU should only be used if you are certain the source and target character sets are compatible. If you are not sure whether PASSTHRU is appropriate in your environment, contact Oracle Support before using it.

OVERRIDE
Forces Replicat to use the specified character set thus overriding the source database character set in the trail file. This option overrides character type column character set except in the following cases:

- The character set is overridden by the CHARSET and COLCHARSET parameters.
- Use of NCHAR, NVARCHAR and NCLOB data types.
- The database overrides the column character set explicitly to a set other than the database character set.

DB2 for z/OS
Valid for DB2 for z/OS.
Required if the version of a trail that contains DB2 data from the z/OS platform is Oracle GoldenGate 12.1 or lower. This parameter ensures that Replicat recognizes that the data is from DB2 for z/OS, which permits a mix of ASCII and EBCDIC character formats.

Examples

Example 1

```
SOURCECHARSET ISO-8859-9
```

Example 2

```
SOURCECHARSET PASSTHRU
```

Example 3

```
SOURCECAHRSET JA16EUC
```

Example 4

```
SOURCECHARSET OVERRIDE WE8ISO8859P15
```

2.145 SOURCEDEFS

Valid For

Extract data pump and Replicat

Description

Use the SOURCEDEFS parameter to specify the name of a file that contains definitions of source tables or files. Source definitions are not required, by default, when trail files with format Oracle GoldenGate release 12.2.*x* are used because the trail files contains metadata records with the object definitions. However, source definitions are required

when replicating data between heterogenous source and targets using trail files with format Oracle GoldenGate release 12.1.*x* and lower or when trail files with created with the `no_objectdefs` option.

Use `SOURCEDEFS` for one or more of the following processes, depending on your Oracle GoldenGate configuration:

- A Replicat process on the target system
- A data pump on a source or intermediary system.

To generate the source-definitions file, use the DEFGEN utility. Transfer the file to the intermediary or target system before starting a data pump or Replicat.

You can have multiple `SOURCEDEFS` statements in the parameter file if more than one source-definitions file will be used, for example if each `SOURCEDEFS` file holds the definitions for a distinct application.

See "ASSUMETARGETDEFS" for related information. Do not use `SOURCEDEFS` and `ASSUMETARGETDEFS` in the same parameter file.

See *Administering Oracle GoldenGate* for more information about using data-definitions files.

Default

None

Syntax

```
SOURCEDEFS file_name [OVERRIDE]
```

file_name
The relative or fully qualified name of the file containing the source data definitions.

OVERRIDE
By default, the table definitions from the metadata records override the definitions from any `SOURCEDEFS` file.
Specify `OVERRIDE` to request Replicat to use the definitions from the definitions file instead of the metadata.

Examples

Example 1

```
SOURCEDEFS dirdef\tcust.def
```

Example 2

```
SOURCEDEFS /ggs/dirdef/source_defs
```

2.146 SOURCEISTABLE

Valid For

Extract

Description

Use the SOURCEISTABLE parameter to extract complete records directly from source tables in preparation for loading them into another table or file. SOURCEISTABLE extracts all column data specified within a TABLE statement.

This parameter applies to the following initial load methods:

- Loading data from file to Replicat.
- Loading data from file to database utility.

Do not use this parameter for the following initial load methods:

- An Oracle GoldenGate direct load, where Extract sends load data directly to the Replicat process without use of a file.
- An Oracle GoldenGate direct bulk load to SQL*Loader.

For those processes, SOURCEISTABLE is specified as an ADD EXTRACT argument instead of being used in the parameter file.

When used, SOURCEISTABLE must be the first parameter statement in the Extract parameter file.

To use SOURCEISTABLE, disable DDL extraction and replication by omitting the DDL parameter from the Extract and Replicat parameter files. See "DDL" for more information.

See *Administering Oracle GoldenGate* for more information about configuring initial data loads.

Default

None

Syntax

```
SOURCEISTABLE
```

2.147 SOURCETIMEZONE

Valid For

Replicat

Description

Use the SOURCETIMEZONE parameter to specify the time zone of the source database. Use this parameter for one of the following purposes:

- To override the source time zone that is stored in the trail. By default, Replicat sets its session to the specified time zone, in both region ID and offset value. This option applies to Oracle GoldenGate versions 12.1.2 or later, where the source time zone is written to the trail by Extract. Replicat will set its session to the specified time zone.
- To supply the time zone of the source database when the trail is written by an Extract version that is older than 12.1.2. In these versions, Extract does not write

the source time zone to the trail, so it must be supplied by this parameter. Replicat will set its session to the specified time zone.

To disable the default use of the source time zone by Replicat, use the PRESERVETARGETTIMEZONE parameter in the Replicat parameter file. See PRESERVETARGETTIMEZONE for more information.

Default

None

Syntax

```
SOURCETIMEZONE time_zone
```

time_zone
The time zone of the source database as output by the database for DATE, TIME and TIMESTAMP data types. It can be specified in the following ways.

- As a region ID that is valid in the IANA Time Zone Database (tz database). (A region ID is also known as an Olson time zone ID). An adjustment for Daylight Saving Time can be performed by the target database, if supported.
- As an offset from UTC.

Examples

The following examples show different ways to specify SOURCETIMEZONE.

- These examples specify a region ID.

```
SOURCETIMEZONE America/New_York

SOURCETIMEZONE US/Pacific

SOURCETIMEZONE Japan

SOURCETIMEZONE UTC

SOURCETIMEZONE Pacific/Guam
```

- These examples specify an offset from UTC.

```
SOURCETIMEZONE +09:00

SOURCETIMEZONE -04:30
```

2.148 SPACESTONULL | NOSPACESTONULL

Valid For

Replicat on Oracle Database only

Description

Use the SPACESTONULL and NOSPACESTONULL parameters to control whether or not a source column that contains only spaces is converted to NULL in the target column. SPACESTONULL converts spaces to NULL if the target column accepts NULL values. NOSPACESTONULL converts spaces to a single space character in the target column.

This parameter is applicable to the follow two scenarios:

- a source column that contains only spaces
- a source column is empty, such as empty CHAR/VARCHAR column data from DB2

Oracle does not distinguish empty and NULL column though other databases do so you should consult your database documentation to determine how these types of columns.

The parameters are table specific. One parameter applies to all subsequent MAP statements, until the other parameter is encountered.

Because you can selectively enable or disable these parameters between MAP statements, you can enable or disable them for different threads of a coordinated Replicat. Specify the SPACESTONULL threads in one set of MAP statements, and specify the NOSPACESTONULL threads in a different set of MAP statements.

Default

NOSPACESTONULL

Syntax

SPACESTONULL | NOSPACESTONULL

Example

This example shows how you can apply SPACESTONULL and NOSPACESTONULL selectively to different MAP statements, each of which represents a different thread of a coordinated Replicat.

```
SPACESTONULL
MAP sales.cust, TARGET sales.cust, THREAD (1);
MAP sales.ord, TARGET sales.ord, THREAD (2);
NOSPACESTONULL
MAP sales.loc, TARGET sales.loc, THREAD (3);
```

2.149 SPECIALRUN

Valid For

Replicat

Description

Use the SPECIALRUN parameter in a Replicat parameter file for a one-time processing run to direct Replicat not to create checkpoints. A one-time run has a beginning and an end, so checkpoints are not needed. Use SPECIALRUN for certain initial data load methods.

When Replicat is in SPECIALRUN mode, do not start it with the START REPLICAT command in GGSCI. It is started automatically during the initial load.

SPECIALRUN requires the use of the END parameter. Either REPLICAT or SPECIALRUN is required in the Replicat parameter file. See "REPLICAT" for more information.

Default

None

Syntax

```
SPECIALRUN
```

2.150 SPLIT_TRANS_REC

Valid For

Parallel Replicat

Description

Specifies that large transactions should be broken into pieces of specified size and applied in parallel. Dependencies between pieces are still honored. Disabled by default..

Syntax

```
SPLIT_TRANS_RECS value
```

Example

```
SPLIT_TRANS_RECS 1000
```

2.151 SQLDUPERR

Valid For

Replicat

Description

Use the `SQLDUPERR` parameter to specify the numeric error code returned by the target database when a duplicate row is encountered. A duplicate-record error indicates that an `INSERT` operation was attempted with a primary key that matches the key of an existing record in the database.

You must use `SQLDUPERR` when you specify the special handling of duplicate records with the `OVERRIDEDUPS` parameter. See "OVERRIDEDUPS | NOOVERRIDEDUPS" for more information.

Default

None

Syntax

```
SQLDUPERR error_number
```

error_number
The numeric error code to return for duplicate records.

Example

```
SQLDUPERR -2601
```

2.152 SQLEXEC

Valid For

Extract and Replicat

Description

Use the SQLEXEC parameter to execute a stored procedure, query, or database command within the context of Oracle GoldenGate processing. SQLEXEC enables Oracle GoldenGate to communicate directly with the database to perform any work that is supported by the database. This work can be part of the synchronization process, such as retrieving values for column conversion, or it can be independent of extracting or replicating data, such as executing a stored procedure that executes an action within the database.

> **Note:**
>
> SQLEXEC provides minimal globalization support. To use SQLEXEC in the capture parameter file of the source capture, make sure that the client character set in the source .prm file is either the same or a superset of the source database character set.

SQLEXEC works as follows:

- As a standalone statement at the root level of a parameter file to execute a SQL stored procedure or query or to execute a database command. As a standalone statement, SQLEXEC executes independently of a TABLE or MAP statement during Oracle GoldenGate processing. When used in a standalone SQLEXEC parameter, a query or procedure cannot include parameters. See "Standalone SQLEXEC".
- As part of a TABLE or MAP parameter to execute a stored procedure or query with or without parameters. When used with parameters, the procedure or query that is executed can accept input parameters from source or target rows and pass output parameters. See "SQLEXEC in a TABLE or MAP Parameter".

> **Caution:**
>
> Use caution when executing SQLEXEC procedures against the database, especially against the production database. Any changes that are committed by the procedure can result in overwriting existing data.

> **Note:**
>
> The SQLEXECONBEFOREIMAGE parameter supports SQLEXEC execution on Before Image records.

Standalone SQLEXEC

A standalone SQLEXEC parameter is one that is used at the root level of a parameter file and acts independently of a TABLE or MAP parameter. The following are guidelines for using a standalone SQLEXEC parameter.

- A standalone SQLEXEC statement executes in the order in which it appears in the parameter file relative to other parameters.
- A SQLEXEC procedure or query must contain all exception handling.
- A query or procedure must be structured correctly when executing a SQLEXEC statement, with legal SQL syntax for the database; otherwise Replicat will abend, regardless of any error-handling rules that are in place. Refer to the SQL reference guide provided by the database vendor for permissible SQL syntax.
- A database credential for the Oracle GoldenGate user must precede the SQLEXEC clause. For Extract, use the SOURCEDB and USERID or USERIDALIAS parameters as appropriate for the database. For Replicat, use the TARGETDB and USERID or USERIDALIAS parameters, as appropriate.
- The database credential that the Oracle GoldenGate process uses is the one that executes the SQL. This credential must have the privilege to execute commands and stored procedures and call database-supplied procedures.
- A standalone SQLEXEC statement cannot be used to get input parameters from records or pass output parameters. You can use stored procedures and queries with parameters by using a SQLEXEC statement within a TABLE or MAP statement. See "SQLEXEC in a TABLE or MAP Parameter".
- All objects affected by a standalone SQLEXEC statement must exist before the Oracle GoldenGate processes start. Because of this, DDL support must be disabled for those objects; otherwise, DDL operations could change the structure of, or delete an object, before the SQLEXEC procedure or query executes on it.
- Object names must be fully qualified in their two-part or three-part name format.
- For DB2 on z/OS, Oracle GoldenGate uses the ODBC SQLExecDirect function to execute a SQL statement dynamically. ODBC prepares the SQL statement every time that it is executed, at a specified interval. To support this function, the connected database server must be configured to prepare SQL dynamically. See the DB2 for z/OS documentation for more information.

Getting More Information about Using Standalone SQLEXEC

See *Administering Oracle GoldenGate* for more information about how to use SQLEXEC.

Syntax for Standalone SQLEXEC

```
SQLEXEC
{'call procedure_name()' | 'SQL_query' | 'database_command'}
[EVERY n {SECONDS | MINUTES | HOURS | DAYS}]
[ONEXIT]
```

[, THREADS (*threadID*[, *threadID*][, ...][, *thread_range*[, *thread_range*]
[, ...])]

'call *procedure_name ()*'
Specifies the name of a stored procedure to execute. Enclose the statement within single quotes. The `call` keyword is required. The following is an example of how to execute a procedure with standalone `SQLEXEC`:

```
SQLEXEC 'call prc_job_count ()'
```

'SQL_query'
Specifies the name of a query to execute. Enclose the query within single quotes. Specify case-sensitive object names in the same format required by the database. The following is an example of how to execute a query with standalone `SQLEXEC`:

```
SQLEXEC ' select x from dual '
```

For a multi-line query, use the single quotes on each line. For best results, type a space after each begin quote and before each end quote (or at least before each end quote).

'database_command'
Executes a database command. The following is an example of how to execute a database command with standalone `SQLEXEC`:

```
SQLEXEC 'SET TRIGGERS OFF'
```

EVERY *n* {SECONDS | MINUTES | HOURS | DAYS}
Causes a standalone stored procedure or query to execute at a defined interval, for example:

```
SQLEXEC 'call prc_job_count ()' EVERY 30 SECONDS
```

The interval must be a whole, positive integer.

ONEXIT
Executes the SQL when the Extract or Replicat process stops gracefully, for example:

```
SQLEXEC 'call prc_job_count ()' ONEXIT
```

**THREADS (*threadID*[, *threadID*][, ...][, *thread_range*[, *thread_range*]
[, ...])**
Executes `SQLEXEC` only for the specified thread or threads of a coordinated Replicat.

> ***threadID*[, *threadID*][, ...]**
> Specifies a thread ID or a comma-delimited list of threads in the format of `threadID, threadID, threadID`.
>
> **[, *thread_range*[, *thread_range*][, ...]**
> Specifies a range of threads in the form of `threadIDlow-threadIDhigh` or a comma-delimted list of ranges in the format of `threadIDlow-threadIDhigh, threadIDlow-threadIDhigh`.

A combination of these formats is permitted, such as `threadID, threadID, threadIDlow-threadIDhigh`.

If no `THREADS` clause is used, the SQL is executed by all of the threads that were configured for this Replicat group by the `ADD REPLICAT` command. However, if the SQL

satisfies the criteria for a barrier transaction, the entire SQLEXEC statement is processed by thread 0 regardless of the actual thread mapping.

SQLEXEC in a TABLE or MAP Parameter

A SQLEXEC parameter in a TABLE or MAP parameter can be used to execute a stored procedure or query that does or does not accept parameters. The following are SQLEXEC dependencies and restrictions when used in a MAP or TABLE statement:

- The SQL is executed by the database user under which the Oracle GoldenGate process is running. This user must have the privilege to execute stored procedures and call database-supplied procedures.
- A query or procedure must be structured correctly when executing a SQLEXEC statement. If Replicat encounters a problem with the query or procedure, the process abends immediately, despite any error-handling rules that are in place. Refer to the SQL reference guide provided by the database vendor for permissible SQL syntax.
- The COMMIT operation of a Replicat transaction to the target database also commits any DML changes that are made in a SQLEXEC statement within the boundary of the original source transaction. This is not true for Extract, because Extract does not perform SQL transactions. When using SQLEXEC for Extract, you can either enable implicit commits or execute an explicit commit within the SQLEXEC procedure.
- Specify literals in single quotes. Specify case-sensitive object names the same way they are specified in the database.
- Do not use SQLEXEC to change the value of a primary key column. The primary key value is passed from Extract to Replicat. Without it, Replicat operations cannot be completed. If primary key values must be changed with SQLEXEC, you may be able to avoid errors by mapping the original key value to another column and then defining that column as a substitute key with the KEYCOLS option of the TABLE and MAP parameters.
- For DB2 on z/OS, Oracle GoldenGate uses the ODBC SQLExecDirect function to execute a SQL statement dynamically. ODBC prepares the SQL statement every time that it is executed, at a specified interval. To support this function, the connected database server must be configured to prepare SQL dynamically. See the DB2 for z/OS documentation for more information.
- When using Oracle GoldenGate to replicate DDL, all objects that are affected by a stored procedure or query must exist with the correct structures prior to the execution of the SQL. Consequently, DDL on these objects that affects structure (such as CREATE or ALTER) must execute before the SQLEXEC executes.
- All object names in a SQLEXEC statement must be fully qualified with their two-part or three-part names, as appropriate for the database.
- Do not use SQLEXEC for tables being processed in pass-through mode by a data-pump Extract group.
- The following data types are supported by SQLEXEC for input and output parameters.
  - Numeric data types
  - Date data types
  - Character data types

- When executed by a coordinated Replicat, SQLEXEC is executed by the thread or threads that are specified with the THREAD or THREADRANGE option of the MAP statement. However, if the SQLEXEC is specified in a MAP parameter that contains the COORDINATED keyword, it is executed as a barrier transaction automatically by the thread with the lowest ID number, regardless of the actual thread mapping.

Getting More Information About Using SQLEXEC in TABLE and MAP

For more information about how to use SQLEXEC, see *Administering Oracle GoldenGate*.

For more information about TABLE and MAP, see "TABLE | MAP".

Syntax for SQLEXEC in TABLE or MAP

```
SQLEXEC (
{SPNAME procedure_name[, ID logical_name] |
   ID logical_name, QUERY ' SQL_query '}
{, PARAMS [OPTIONAL | REQUIRED] parameter_name = {source_column |
OGG_function} |
   NOPARAMS}
[, AFTERFILTER | BEFOREFILTER]
[, ALLPARAMS {OPTIONAL | REQUIRED}]
[, ERROR {IGNORE | REPORT | RAISE | FINAL | FATAL}]
[, EXEC {MAP | ONCE | TRANSACTION | SOURCEROW}][, MAXVARCHARLEN bytes]
[, PARAMBUFSIZE bytes]
[, TRACE]
[, ...]
[, BEFORE_col1 = @BEFORE(col1),
)
```

SPNAME *procedure_name*[, ID *logical_name*]
Executes a stored procedure.

SPNAME *procedure_name*
Specifies the name of the procedure to execute.
The following example shows a single execution of a stored procedure named lookup. In this case, the actual name of the procedure is used. A logical name is not needed.

```
SQLEXEC (SPNAME lookup), PARAMS (param1 = srccol)), &
COLMAP (targcol = lookup.param1);
```

ID *logical_name*
Defines an optional logical name for the procedure. For example, logical names for a procedure named lookup might be lookup1, lookup2, and so forth. Use this option to execute the procedure multiple times within a MAP statement. A procedure can execute up to 20 times per MAP statement. ID is not required when executing a procedure once.
The following example shows the use of the ID option to enable multiple executions of a stored procedure that gets values from a lookup table. The values are mapped to target columns.

```
SQLEXEC (SPNAME lookup, ID lookup1, &
  PARAMS (long_name = current_residence_state)), &
SQLEXEC (SPNAME lookup, ID lookup2, &
```

```
  PARAMS (long_name = birth_state)), &
COLMAP (custid = custid, current_residence_state_long = lookup1.long_name,
& birth_state_long = lookup2.long_name);
```

ID *logical_name*, QUERY ' *SQL_query* '
Executes a query.

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 can either return results with a SELECT statement or execute an INSERT, UPDATE, or DELETE statement. A SELECT statement should only return one row. If multiple rows are returned, only the first row is processed. Do not specify an INTO ... clause for any SELECT statements.The query must be valid, standard query language for the database against which it is being executed. Most queries require placeholders for input parameters. How parameters are specified within the query depends on the database type, as follows:

- For Oracle, input parameters are specified by using a colon (:) followed by the parameter name, as in the following example.

  ```
  'SELECT NAME FROM ACCOUNT WHERE SSN = :SSN AND ACCOUNT = :ACCT'
  ```

- For other databases, input parameters are specified by using a question mark, as in the following example.

  ```
  'SELECT NAME FROM ACCOUNT WHERE SSN = ? AND ACCOUNT = ?'
  ```

The query must be contained on one line, within single quotes. Quotation marks are not required around a parameter name for any database.
The following examples illustrate the use of a SQLEXEC query for Oracle and SQL Server queries, respectively.
Oracle example:

```
MAP sales.account, TARGET sales.newacct, &
  SQLEXEC (ID lookup, &
  QUERY 'select desc_col into desc_param from lookup_table &
  where code_col = :code_param', &
  PARAMS (code_param = account_code)), &
  COLMAP (newacct_id = account_id, newacct_val = lookup.desc_param);
```

SQL Server example:

```
MAP sales.account, TARGET sales.newacct, &
  SQLEXEC (ID lookup, &
  QUERY 'select desc_col into desc_param from lookup_table &
  where code_col = ?', &
  PARAMS (p1 = account_code)), &
  COLMAP (newacct_id = account_id, &
  newacct_val = lookup.desc_param);
```

PARAMS [OPTIONAL | REQUIRED] *parameter_name* = {*source_column* | *OGG_function*} |
NOPARAMS
Defines whether or not the procedure or query accepts parameters and, if yes, maps the parameters to the input source. Either a PARAMS clause or NOPARAMS must be used.

OPTIONAL | REQUIRED
Determines whether or not the procedure or query executes when parameter values are missing.
OPTIONAL indicates that a parameter value is not required for the SQL to execute. If a required source column is missing from the database operation, or if a column-conversion function cannot complete successfully because a source column is missing, the SQL executes anyway. OPTIONAL is the default for all databases except Oracle. For Oracle, whether or not a parameter is optional is automatically determined when retrieving the stored procedure definition.
REQUIRED indicates that a parameter value must be present. If the parameter value is not present, the SQL will not be executed.

***parameter_name* = {*source_column* | *OGG_function*}**
Maps the name of a parameter to a column or function that provides the input. The following data types are supported by SQLEXEC for input and output parameters.

- Numeric data types
- Date data types
- Character data types

parameter_name is one of the following:

- For a stored procedure, it is the name of any parameter in the procedure that can accept input.
- For an Oracle query, it is the name of any input parameter in the query *excluding* the leading colon. For example, :vemplid would be specified as vemplid in the PARAMS clause. Oracle permits naming an input parameter any logical name.

```
SQLEXEC (ID appphone, QUERY ' select per_type from ps_personal_data '
    ' where emplid = :vemplid '
    ' and per_status = 'N' and per_type = 'A' ',
    PARAMS (vemplid = emplid)),
TOKENS (applid = @GETVAL(appphone.per_type));
```

- For a non-Oracle query, it is P*n*, where *n* is the number of the parameter within the statement, starting from 1. For example, in a query with two parameters, the *parameter_name* entries are p1 and p2. Consider whether the database requires the p to be upper or lower case.

```
SQLEXEC (ID appphone, QUERY ' select per_type from ps_personal_data '
    ' where emplid = ? '
    ' and per_status = 'N' and per_type = 'A' ',
    PARAMS (p1 = emplid)),
TOKENS (applid = @GETVAL(appphone.per_type));
```

source_column is the name of a source column that provides the input. By default, if the specified column is not present in the log (because the record only contains the values of columns that were updated) the parameter assumes any default value specified by the procedure or query for the parameter.
OGG_function is the name of an Oracle GoldenGate column-conversion function that executes to provide the input. See "Column Conversion Functions".

To pass output values from the stored procedure or query as input to a FILTER or COLMAP clause, use the following syntax:

```
{procedure_name | logical_name}.parameter
```

Where:

- *procedure_name* is the actual name of a stored procedure, which must match the value given for SPNAME in the SQLEXEC statement. Use this argument only if executing a procedure one time during the course of the Oracle GoldenGate run.
- *logical_name* is the logical name specified with the ID option of SQLEXEC. Use this argument to pass input values from either a query or an instance of a stored procedure when the procedure executes multiple times within a MAP statement.
- *parameter* is the name of a parameter or RETURN_VALUE if extracting returned values. By default, output values are truncated at 255 bytes per parameter. If output parameters must be longer, use the MAXVARCHARLEN option.

> **Note:**
>
> As an alternative to the preceding syntax, you can use the @GETVAL function. See "GETVAL" for more information.

The following examples apply to a set of Oracle source and target tables and a lookup table. These examples show how parameters for the tables are passed for a single instance of a stored procedure and multiple instances of a stored procedure.
Source table cust:

```
custid                    Number
current_residence_state   Char(2)
birth_state               Char(2)
```

Target table cust_extended:

```
custid                          Number
current_residence_state_long    Varchar(30)
birth_state_long                Varchar(30)
```

Lookup table state_lookup

```
abbreviation     Char(2)
long_name        Varchar(30)
```

The following example shows the use of a stored procedure that executes once to get a value from the lookup table. When processing records from the cust table, Oracle GoldenGate executes the lookup stored procedure before executing the column map. The long_name parameter in the procedure accepts input from the birth_state source column.The value is mapped to the target column birth_state_long in the COLMAP statement.

```
MAP sales.cust, TARGET sales.cust_extended, &
SQLEXEC (SPNAME lookup, &
PARAMS (long_name = birth_state)), &
COLMAP (custid = custid, &
birth_state_long = lookup.long_name);
```

The following example shows the use of the ID option to enable multiple executions of a stored procedure that gets values from a lookup table. The values are mapped to target columns.

```
MAP sales.cust, TARGET sales.cust_extended, &
SQLEXEC (SPNAME lookup, ID lookup1, &
PARAMS (long_name = current_residence_state)), &
SQLEXEC (SPNAME lookup, ID lookup2, &
PARAMS (long_name = birth_state)), &
COLMAP (custid = custid, current_residence_state_long = lookup1.long_name, &
birth_state_long = lookup2.long_name);
```

AFTERFILTER | BEFOREFILTER
Use AFTERFILTER and BEFOREFILTER to specify when to execute the stored procedure or query in relation to the FILTER clause of a MAP statement.

AFTERFILTER
Causes the SQL to execute after the FILTER statement. This enables you to skip the overhead of executing the SQL unless the filter is successful. This is the default.

BEFOREFILTER
Causes the SQL to execute before the FILTER statement, so the results can be used in the filter.

The following is an example using BEFOREFILTER.

```
SQLEXEC (SPNAME check, NOPARAMS, BEFOREFILTER)
```

ALLPARAMS [OPTIONAL | REQUIRED]
Use ALLPARAMS as a global rule that determines whether or not all of the specified parameters must be present for the stored procedure or query to execute. Rules for individual parameters established within the PARAMS clause override the global rule set with ALLPARAMS.

OPTIONAL
Permits the SQL to execute whether or not all of the parameters are present. This is the default.

REQUIRED
Requires all of the parameters to be present for the SQL to execute.

The following is an example using OPTIONAL.

```
SQLEXEC (SPNAME lookup,
PARAMS (long_name = birth_state, short_name = state),
ALLPARAMS OPTIONAL)
```

ERROR {IGNORE | REPORT | RAISE | FINAL | FATAL}
Use ERROR to define a response to errors associated with the stored procedure or query. Without explicit error handling, the Oracle GoldenGate process abends on errors. Make certain your procedures return errors to the process and specify the responses with ERROR.

IGNORE
Causes Oracle GoldenGate to ignore all errors associated with the stored procedure or query and continue processing. Any resulting parameter extraction results in "column missing" conditions. This is the default.

REPORT
Ensures that all errors associated with the stored procedure or query are reported to the discard file. The report is useful for tracing the cause of the error. It includes both an error description and the value of the parameters passed to and from the procedure or query. Oracle GoldenGate continues processing after reporting the error.

RAISE
Handles errors according to rules set by a REPERROR parameter. Oracle GoldenGate continues processing other stored procedures or queries associated with the current MAP statement before processing the error.

FINAL
Is similar to RAISE except that when an error associated with a procedure or query is encountered, remaining stored procedures and queries are bypassed. Error processing is invoked immediately after the error.

FATAL
Causes Oracle GoldenGate to abend immediately upon encountering an error associated with a procedure or query.

EXEC {MAP | ONCE | TRANSACTION | SOURCEROW}
Use EXEC to control the frequency with which a stored procedure or query in a MAP statement executes and how long the results are considered valid, if extracting output parameters.

MAP
Executes the procedure or query once for each source-target table map for which it is specified. Using MAP renders the results invalid for any subsequent maps that have the same source table. MAP is the default.
The following example shows the incorrect use of the default of MAP. Because MAP is the default, it need not be explicitly listed in the SQLEXEC statement.
In this example, a source table is mapped in separate MAP parameters to two different target tables. In this case, the results are valid only for the first mapping. The results of the procedure lookup are expired by the time the second MAP parameter executes, and the second MAP results in a "column missing" condition. To implement this correctly so that each MAP returns valid results, SOURCEROW should be used.

```
MAP sales.srctab, TARGET sales.targtab, &
SQLEXEC (SPNAME lookup, PARAMS (param1 = srccol)), &
COLMAP (targcol = lookup.param2);

MAP sales.srctab, TARGET sales.targtab2, &
COLMAP (targcol2 = lookup.param2);
```

ONCE
Executes the procedure or query once during the course of the Oracle GoldenGate run, upon the first invocation of the associated MAP statement. The results remain valid for as long as the process remains running.
The following is an example of using ONCE.

```
MAP sales.cust, TARGET sales.cust_extended, &
SQLEXEC (SPNAME lookup, PARAMS (long_name = birth_state), EXEC ONCE), &
COLMAP (custid = custid, &
birth_state_long = lookup.long_name);
```

TRANSACTION
Executes the procedure or query once per source transaction. The results remain valid for all operations of the transaction.
The following is an example of using TRANSACTION.

```
MAP sales.cust, TARGET sales.cust_extended, &
SQLEXEC (SPNAME lookup, PARAMS (long_name = birth_state), EXEC
TRANSACTION), &
COLMAP (custid = custid, &
birth_state_long = lookup.long_name);
```

SOURCEROW
Executes the procedure or 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.
The following is an example of using SOURCEROW. In this case, the second map returns a valid value because the procedure executes on every source row operation.

```
MAP sales.srctab, TARGET sales.targtab, &
SQLEXEC (SPNAME lookup, PARAMS (param1 = srccol), EXEC SOURCEROW), &
COLMAP (targcol = lookup.param2);

MAP sales.srctab, TARGET sales.targtab2, &
COLMAP (targcol2 = lookup.param2);
```

MAXVARCHARLEN *bytes*
Use MAXVARCHARLEN to specify the maximum byte length allocated for the output value of any parameter in a stored procedure or query. Beyond this maximum, the output values are truncated. The default is 255 bytes without an explicit MAXVARCHARLEN clause. The valid range of values is from 50 to 32767 bytes.
The following example limits the byte length of output values to 100.

```
MAXVARCHARLEN 100
```

PARAMBUFSIZE *bytes*
Use PARAMBUFSIZE to specify the maximum number of bytes allowed for the memory buffer that stores SQLEXEC parameter information, including both input and output parameters. The default is 10,000 bytes without an explicit PARAMBUFSIZE clause. The valid range of values is from 1000 to 2000000 bytes. Oracle GoldenGate issues a warning whenever the memory allocated for parameters is within 500 bytes of the maximum.
The following example increases the buffer to 15,000 bytes.

```
PARAMBUFSIZE 15000
```

TRACE {ALL | ERROR}
Use TRACE to log SQLEXEC input and output parameters to the report file.
The following is a sample report file with SQLEXEC tracing enabled:

```
Input parameter values...
LMS_TABLE: INTERACTION_ATTR_VALUES
   KEY1: 2818249
  KEY2: 1
Report File:
From Table MASTER.INTERACTION_ATTR_VALUES to MASTER.INTERACTION_ATTR_VALUES:
       #  inserts:      0
```

```
    #  updates:      0
    #  deletes:      0
    #  discards:     1

Stored procedure GGS_INTERACTION_ATTR_VALUES:
    attempts:        2
    successful:      0
```

ALL
Writes the input and output parameters for each invocation of the procedure or query to the report file. This is the default.

ERROR
Writes the input and output parameters for each invocation of the procedure or query to the report file only after a SQL error occurs.

2.153 STATOPTIONS

Valid For

Extract and Replicat

Description

Use the `STATOPTIONS` parameter to specify the information that is to be included in statistical displays generated by the `STATS EXTRACT` or `STATS REPLICAT` command. These options also can be enabled as needed as arguments to those commands.

Default

See individual options.

Syntax

```
STATOPTIONS
[, REPORTDETAIL | NOREPORTDETAIL]
[, REPORTFETCH | NOREPORTFETCH]
[, RESETREPORTSTATS | NORESETREPORTSTATS]
[, THREADS (threadID[, threadID][, ...][, thread_range[, thread_range]
[, ...])]
```

REPORTDETAIL | NOREPORTDETAIL
Valid for Replicat. Controls the reporting of statistics for operations that were not applied to the target because they were discarded as the result of collision handling.

REPORTDETAIL
Returns statistics for the discarded operations. These operations are reported in the regular `STATS REPLICAT` statistics (inserts, updates, and deletes performed) and as discard statistics if `STATS REPLICAT` is issued with the `DETAIL` option. For example, if 10 records were `INSERT` operations and they were all ignored due to duplicate keys, the report would indicate that there were 10 inserts and also 10 discards due to collisions. `REPORTDETAIL` is the default.

NOREPORTDETAIL
Turns off the reporting of statistics for discarded operations.

REPORTFETCH | NOREPORTFETCH
Valid for Extract. Controls the reporting of statistics for the amount of row fetching performed by Extract, such as the fetches that are triggered by a FETCHCOLS clause or fetches that must be performed when not enough information is in the transaction record.

REPORTFETCH
Reports statistics for row fetching. The output is as follows:

- `row fetch attempts`: The number of times Extract attempted to fetch a column value from the database when it could not obtain the value from the transaction log.
- `fetch failed`: The number of `row fetch attempts` that failed.
- row fetch by key: (Valid for Oracle) The number of row fetch attempts that were made by using the primary key.

NOREPORTFETCH
Turns off the reporting of fetch statistics. NOREPORTFETCH is the default.

RESETREPORTSTATS | NORESETREPORTSTATS
Valid for Extract and Replicat. Controls whether or not statistics generated by the REPORT parameter are reset when a new report is created. RESETREPORTSTATS resets the statistics from one report to the other. NORESETREPORTSTATS continues the statistics from one report to another and is the default, see REPORT. Report rollover is controlled by the REPORTROLLOVER parameter, see REPORTROLLOVER.

THREADS (*threadID*[, *threadID*][, ...][, *thread_range*[, *thread_range*][, ...])
Enables the selected STATOPTIONS options for the specified threads of a coordinated Replicat.

***threadID*[, *threadID*][, ...]**
Specifies a thread ID or a comma-delimited list of threads in the format of `threadID, threadID, threadID`.

[, *thread_range*[, *thread_range*][, ...]
Specifies a range of threads in the form of `threadIDlow-threadIDhigh` or a comma-delimited list of ranges in the format of `threadIDlow-threadIDhigh, threadIDlow-threadIDhigh`.

A combination of these formats is permitted, such as `threadID, threadID, threadIDlow-threadIDhigh`.

Examples

This example includes fetch details of a coordinated Replicat.

```
STATOPTIONS REPORTFETCH
```

This example resets the statistics from one report to another for thread 0 of a coordinated Replicat .

```
STATOPTIONS RESETREPORTSTATS THREADS 0
```

2.154 TABLE | MAP

Valid For

TABLE is valid for Extract. MAP is valid for Replicat

Description

The TABLE and MAP parameters control the selection, mapping, and manipulation of the objects that are to be affected by an Oracle GoldenGate process. These parameters work as follows:

- Use the TABLE parameter in an Extract parameter file to specify one or more objects that are to be captured from the data source by the Extract process. TABLE options specify processing work such as filtering and token definitions that must be performed before Extract writes the captured data to the Oracle GoldenGate trail.
- Use the MAP parameter in the Replicat parameter file to map the data from the source objects to the appropriate target objects. MAP options specify processing work such as filtering, conversion, and error handling that must be performed before the data is applied to the target objects. Each target object that you want to synchronize with a source object must be associated with that source object by means of a MAP parameter. Multiple source-target relationships can be specified by means of a wildcard.

TABLE and MAP are valid for initial load configurations and for online processes configured to support the replication of transactional changes.

You can process the following objects with TABLE and MAP:

- Indexes
- Triggers
- Materialized views
- Tables

To specify a sequence for capture by Extract, use the SEQUENCE parameter.

> **Note:**
>
> Oracle GoldenGate supports the replication of the actual data values of Oracle materialized views. Oracle GoldenGate supports the replication of Oracle and Teradata DDL for indexes and triggers, but not the content of those objects.

You can use one or more TABLE or MAP statements in a parameter file, with or without wildcards, to specify all of the objects that you want to process.

You can exclude objects from a wildcarded TABLE or MAP statement with the TABLEEXCLUDE and MAPEXCLUDE parameters. Additional exclusion parameters are CATALOGEXCLUDE, SCHEMAEXCLUDE, and EXCLUDEWILDCARDOBJECTSONLY.

For more information about using TABLE and MAP, see *Administering Oracle GoldenGate*.

Default

None

Syntax for TABLE

For tables, you can use all of the TABLE options. For non-table objects, use TABLE only to specify an object for capture.

```
TABLE source_table[, TARGET target_table]
[, ATTRCHARSET (charset)]
[, CHARSET character_set]
[, COLCHARSET character_set]
[, COLMAP (column_mapping)]
[, {COLS | COLSEXCEPT} (column_list)]
[, {DEF | TARGETDEF} template]
[, EVENTACTIONS action]
[, EXITPARAM 'parameter']
[, {FETCHCOLS | FETCHCOLSEXCEPT} (column_list)]
[, {FETCHMODCOLS | FETCHMODCOLSEXCEPT} (column_list)]
[, FETCHBEFOREFILTER]
[, FILTER (filter_clause)]
[, GETBEFORECOLS (column_specification)]
[, KEYCOLS (columns)]
[, SQLEXEC (SQL_specification)]
[, SQLPREDICATE 'WHERE where_clause']
[, TOKENS (token_definition)]
[, TRIMSPACES | NOTRIMSPACES]
[, TRIMVARSPACES | NOTRIMVARSPACES]
[, WHERE (clause)]
[, container.]schema.table PARTITIONOBJID ptn_object_ID [, ptn_object_ID]
;
```

Syntax for MAP

```
MAP source_table, TARGET target_table
[, MOD_COMPARE_COLS( tgt_col = source )]
[, COLMAP (column_mapping)]
[, COMPARECOLS (column_specification)]
[, COORDINATED]
[, {DEF | TARGETDEF} template]
[, EXCEPTIONSONLY]
[, EXITPARAM 'parameter']
[, EVENTACTIONS (action)]
[, FILTER (filter_clause)]
[, HANDLECOLLISIONS | NOHANDLECOLLISIONS]
[, INSERTALLRECORDS]
[, INSERTAPPEND | NOINSERTAPPEND]
[, KEYCOLS (columns)]
[, MAPALLCOLUMNS | NOMAPALLCOLUMNS]
[, MAPEXCEPTION (exceptions_mapping)]
[, MAPINVISIBLECOLUMNS | NOMAPINVISIBLECOLUMNS]
[, REPERROR (error, response)]
[, RESOLVECONFLICT (conflict_resolution_specification)]
[, SQLEXEC (SQL_specification)]
[, THREAD (thread_ID)]
[, THREADRANGE (thread_range[, column_list])]
```

```
[, TRIMSPACES | NOTRIMSPACES]
[, TRIMVARSPACES | NOTRIMVARSPACES]
[, WHERE (clause)]
[, container.]schema.table PARTITIONOBJID ptn_object_ID [, ptn_object_ID]
;
```

TABLE and MAP Options

The following table summarizes the options that are available for the TABLE and MAP parameters. Note that not all options are valid for both parameters.

Table 2-14 Summary of TABLE and MAP Syntax Components

Component	Description	Valid For
MAP MOD_COMPARE_COL(tgt_col = source [,...])	This is a Replicat only parameter. Assigns specified source value to target column's before image as key value, and the value is used for WHERE clause.	MAP
TABLE *source_table*[, TARGET *taget_table*]	Specifies the source object in a TABLE statement for Extract and an optional mapping to a target object. Use in the Extract parameter file.	TABLE
MAP *source_table*, TARGET *target_table*	Specifies the source-target object mapping for the Replicat process. Use in the Replicat parameter file.	MAP
ATTRCHARSET *(charset)*	specifies the source character set information at UDT attribute level.	TABLE
CHARSET *character_set*	Specifies any supported character set.	TABLE
COLCHARSET *character_set*	Specifies any supported character set.	TABLE
COLMAP *(column_mapping)*	Maps records between different source and target columns.	TABLE and MAP

Table 2-14 (Cont.) Summary of TABLE and MAP Syntax Components

Component	Description	Valid For
{COLS | COLSEXCEPT} (*column_list*)	Selects or excludes columns for processing.	TABLE
COMPARECOLS (*column_specification*)	Specifies columns to use for conflict detection and resolution.	TABLE and MAP
COORDINATED	Forces a transaction to be processed as a barrier transaction.	MAP
{DEF| TARGETDEF} *template*	Specifies a source-definitions or target-definitions template.	TABLE and MAP
EXCEPTIONSONLY	Specifies that the MAP statement is an exceptions MAP statement.	MAP
EVENTACTIONS (*action*)	Triggers an action based on a record that satisfies a specified filter rule.	TABLE and MAP
EXITPARAM '*parameter*'	Passes a parameter in the form of a literal string to a user exit.	TABLE and MAP
FETCHBEFOREFILTER	Directs the FETCHCOLS or FETCHCOLSEXCEPT action to be performed before a filter is executed.	TABLE
{FETCHCOLS | FETCHCOLSEXCEPT} (*column_list*)	Enables the fetching of column values from the source database when the values are not in the transaction record.	TABLE
{FETCHMODCOLS | FETCHMODCOLSEXCEPT} (*column_list*)	Forces column values to be fetched from the database when the columns are present in the transaction log.	TABLE

Table 2-14 (Cont.) Summary of TABLE and MAP Syntax Components

Component	Description	Valid For
FILTER (*filter_clause*)	Selects records based on a numeric value. FILTER provides more flexibility than WHERE.	TABLE and MAP
GETBEFORECOLS (*column_specification*)	Forces before images of columns to be captured and written to the trail.	TABLE
HANDLECOLLISIONS | NOHANDLECOLLISIONS	Reconciles the results of changes made to the target table by an initial load process with those applied by a change-synchronization group.	MAP
INSERTALLRECORDS	Applies all row changes as inserts.	MAP
INSERTAPPEND | NOINSERTAPPEND	Controls whether or not Replicat uses an Oracle APPEND hint for INSERT statements.	MAP
KEYCOLS (*columns*)	Designates columns that uniquely identify rows.	TABLE and MAP
MAPALLCOLUMNS| NOMAPALLCOLUMNS	Controls whether or not Replicat obtains non-key columns.	
MAPEXCEPTION (*exceptions_mapping*)	Specifies that the MAP statement contains exceptions handling for wildcarded tables.	MAP

Table 2-14 (Cont.) Summary of TABLE and MAP Syntax Components

Component	Description	Valid For
`MAPINVISIBLECOLUMNS | NOMAPINVISIBLECOLUMNS`	Controls whether or not Replicat includes invisible columns in Oracle target tables for default column mapping. For invisible columns in Oracle target tables that use explicit column mapping, they are always mapped so do not require this option.	MAP
`REPERROR (error, response)`	Controls how Replicat responds to errors when executing the MAP statement.	MAP
`RESOLVECONFLICT (conflict_resolution_specification)`	Specifies rules for conflict resolution.	MAP
`SQLEXEC (SQL_specification)`	Executes stored procedures and queries.	TABLE and MAP
`SQLPREDICATE 'WHERE where_clause'`	Enables a WHERE clause to select rows for an initial load.	TABLE
`THREAD (thread_ID)`	Valid for Replicat in coordinated mode. Specifies that the MAP statement will be processed by the specified Replicat thread.	MAP
`THREADRANGE (thread_range, column_list)`	Valid for Replicat in coordinated mode. Specifies that the MAP statement will be processed by the specified range of Replicat threads.	MAP
`TOKENS (token_definition)`	Defines user tokens.	TABLE

Table 2-14 (Cont.) Summary of TABLE and MAP Syntax Components

Component	Description	Valid For
TRIMSPACES | NOTRIMSPACES	Controls whether trailing spaces are trimmed or not when mapping CHAR to VARCHAR columns.	TABLE and MAP
TRIMVARSPACES | NOTRIMVARSPACES	Controls whether trailing spaces are trimmed or not when mapping VARCHAR to CHAR or VARCHAR columns.	TABLE and MAP
WHERE (*clause*)	Selects records based on conditional operators.	TABLE and MAP
;	(Semicolon) Terminates the TABLE or MAP statement and is required.	TABLE and MAP
PARTITIONOBJID	Available for Integrated Extract. This option is used to specify the object IDs of the partitions to be captured for partitioned tables.	TABLE and MAP

MAP MOD_COMPARE_COL(tgt_col = source [,...])

tgt_col must be target table column name, and should be the key column to take effect properly. source can be source table column, constant value (number or string), column mapping function or SQLEXEC results. For example, source col1 is mapped to target col1. source col1 before image value is 1, which is a dummy value because it is masked by DBA for security. Replicat can query actual before image value of target col1 by SQLEXEC. Using MOD_COMPARE_COLS(), Replicat replaces dummy source col1 value by SQLEXEC result, so that UPDATE or DELETE operation works properly.

TABLE *source_table*[, TARGET *taget_table*]

TABLE is valid in an Extract parameter file.

Use TABLE to specify a source object for which you want Extract to capture data. Specify the fully qualified two-part or three-part name of the object, such as *schema.table* or *catalog.schema.table*. You can use a wildcard to specify multiple

objects with one TABLE statement. To specify object names and wildcards correctly, see *Administering Oracle GoldenGate*.

Use the TARGET option only when Extract must refer to a target definitions file (specified with the TARGETDEFS parameter) to perform conversions or when the COLMAP option is used to map columns. Otherwise, it can be omitted from a TABLE parameter. Column mapping with COLMAP and conversion work usually are performed on the target system to minimize the impact of replication activities on the source system, but can be performed on the source system if required. For example, column mapping and conversion can be performed on the source system in a configuration where there are multiple sources and one target. In this scenario, it may be easier to manage one target definitions file rather than managing a definitions file for each source database, especially if there are frequent application changes that require new definitions files to be generated.

Using TARGET in a TABLE parameter identifies the metadata of the extracted data based on the target structure, rather than that of the source, to reflect the structure of the record that is reflected in the definitions file or the column map. Do not use three-part names if TARGET specifies tables in a target Oracle container database. Replicat can only connect to one container or catalog, so it is assumed that the container or catalog portion of the name is the same as the one that Replicat logs into (as specified with USERID, USERIDALIAS, or TARGETDB, depending on the database).

If no other TABLE syntax options are required to process the specified source data, you can use a simple TABLE statement, making sure to terminate it with a semicolon.

```
TABLE sales.customers;
```

The following shows the use of a wildcard to specify multiple tables:

```
TABLE sales.*;
```

The preceding TABLE statements direct Extract to capture all supported column data for the specified objects and write it to the trail without performing any filtering, conversion, or other manipulation.

MAP *source_table*, TARGET *target_table*

MAP is valid in a Replicat parameter file.

Use MAP to specify a source object, and use TARGET to specify the target object to which Replicat applies the replicated source data. Together, the MAP and TARGET clause comprise a *mapping*.

- For MAP *source_table*, specify the source object. Specify the fully qualified two-part or three-part name of the object, such as *schema.table* or *catalog.schema.table*. You can use a wildcard to specify multiple source objects.
- For TARGET *target_table*, specify a two-part name, even if the target is a container database. Replicat can only connect to one container or catalog, so it is assumed that the container or catalog portion of the name is the same as the one that Replicat logs into (as specified with USERID, USERIDALIAS, or TARGETDB, depending on the database). You can use a wildcard to specify multiple target objects.

The following shows the use of a wildcard to specify multiple tables. Note that the TARGET clause does not include the tab prefix before the wildcard. That specification

would be invalid, because the wildcard would be resolved as `sales.tabtab1`, `sales.tabtab2`, and so forth.

```
MAP sales.tab*, TARGET sales.*;
```

To specify object names and wildcards correctly in the `MAP` and `TARGET` clauses, see *Administering Oracle GoldenGate*.

If no filtering, mapping, or other work is required for the objects, you can use simple `MAP` statements like the following, making sure to terminate each one with a semicolon.

```
MAP sales.customers, TARGET sales.customers;
MAP fin.*, TARGET fin.*;
```

`ATTRCHARSET` *`(charset)`*

`ATTRCHARSET` is valid for `TABLE`.

Use the `ATTRCHARSET` clause to specify the source character set information at UDT attribute level. It overrides the character set defined in the trail file or specified by `SOURCECHARSET`, `CHARSET`, or `COLCHARSET` parameters.

Valid values are character set names and valid UDT attribute names. Wildcard attribute names are supported. For example:

```
TABLE SCHEMA.T*,
  ATTRCHARSET(WE8DEC, col*.attr1, col1.attr*.attr3);
```

`CHARSET` *`character_set`*

`CHARSET` is valid for `TABLE`.

Use the `CHARSET` clause to specify any supported character set. See CHARSET for more information.

`COLCHARSET` *`character_set`*

`COLCHARSET` is valid for `TABLE`.

Use the `COLCHARSET` clause to specify any supported character set. See COLCHARSET for more information.

`COLMAP` *`(column_mapping)`*

`COLMAP` is valid for `TABLE` and `MAP`.

Use `COLMAP` to:

- Map individual source columns to target columns when the source and target columns have different names.
- Specify default column mapping when the source and target names are identical.

`COLMAP` provides instructions for selecting, translating, and moving column data from a source column to a target column.

> **Note:**
>
> To create *global* rules for column mapping across all tables in subsequent MAP statements, use the COLMATCH parameter.

Getting More Information About Configuring Column Mapping

See *Administering Oracle GoldenGate* for more information about using COLMAP. To use COLMAP, related configuration considerations must be taken into account, such as whether source and target column structures are identical or different and whether global column mapping parameters may be sufficient.

Syntax

```
COLMAP (
[USEDEFAULTS, ]
target_column = source_expression [BINARYINPUT]
[, ...]
)
```

USEDEFAULTS
Automatically maps source and target columns that have the same name if they were not specified in an explicit column mapping. The data types are translated automatically, as needed, based on the local data-definitions file. USEDEFAULTS eliminates the need for an explicit column mapping if those columns have the same name and the data does not require any filtering or conversion.
Specify USEDEFAULTS before explicit column mappings in the COLMAP clause. For additional information about default column mapping in COLMAP, see *Administering Oracle GoldenGate*.

target_column = source_expression
Defines an explicit source-target column mapping.

target_column
Specifies the name of the target column. For supported characters in column names, see *Administering Oracle GoldenGate*.

source_expression
Can be any of the following:

- The name of a source column, such as ORD_DATE
- A numeric constant, such as 123
- A string constant within single quotes, such as 'ABCD'
- An expression using an Oracle GoldenGate column-conversion function, such as @STREXT (COL1, 1, 3). See "Column Conversion Functions" for more information.

BINARYINPUT
Use BINARYINPUT when the target column is defined as a binary data type, such as RAW or BLOB, but the source input contains binary zeros in the middle of the data. he source input is handled as binary input, and replacement of data values is suppressed.

Example 1

```
MAP ggs.tran, TARGET ggs.tran2, COLMAP (loc2 = loc, type2 = type);
```

Example 2

```
TABLE ggs.tran, COLMAP (SECTION = @STRCAT('\u00a7', SECTION ));
```

{COLS | COLSEXCEPT} (*column_list*)

COLS and COLSEXCEPT are valid for TABLE.

Use COLS and COLSEXCEPT to control the columns for which data is captured.

- COLS specifies columns that contain the data that you want to capture. When COLS is used, all columns that are not in the COLS list are ignored by Oracle GoldenGate.
- COLSEXCEPT specifies columns to exclude from being captured. When COLSEXCEPT is used, all columns that are not in the COLSEXCEPT list are captured by Oracle GoldenGate. For tables with numerous columns, COLSEXCEPT may be more efficient than listing each column with COLS.

> **Caution:**
>
> Do *not* exclude key columns, and do *not* use COLSEXCEPT to exclude columns that contain data types that are not supported by Oracle GoldenGate. COLSEXCEPT does not exclude unsupported data types.

To use COLS, the following is required:

- The table must have one or more key columns, or a substitute key must be defined with the KEYCOLS option. See "KEYCOLS (*columns*)".
- The key columns or the columns specified with KEYCOLS must be included in the column list that is specified with COLS. Otherwise, they will not be captured, and an error will be generated during processing.

Without a primary key, a unique key, or a KEYCOLS clause in the TABLE statement, Oracle GoldenGate uses all of the columns in the table, rendering COLS unnecessary.

> **Note:**
>
> Do not use this option for tables that are processed in pass-through mode by a data-pump Extract group.

Syntax

```
{COLS | COLSEXCEPT} (column [, ...])
```

column
The name of a column. To specify multiple columns, create a comma-delimited list, for example:

```
COLS (name, city, state, phone)
```

> **Note:**
>
> If the database only logs values for columns that were changed in an update operation, a column specified for capture with COLS might not be available. To make those columns available, use the FETCHCOLS option in the TABLE statement or enable supplemental logging for the column.

Example

The COLS clause in this example captures *only* columns 1 and 3, whereas the COLSEXCEPT clause captures all columns *except* columns 1 and 3.

```
TABLE hq.acct, COLS (col1, col3);
TABLE hq.sales, COLSEXCEPT (col1, col3);
```

COMPARECOLS (*column_specification*)

COMPARECOLS is valid for MAP.

Use COMPARECOLS to specify the columns that Replicat uses to detect and resolve update or delete conflicts when configured with the RESOLVECONFLICT option of MAP in a multi-master configuration. A conflict is a mismatch between the before image of a record in the trail and the correct data in the target table.

To use COMPARECOLS, the before image must be available in the trail record by means of the GETBEFORECOLS parameter in the Extract TABLE statement. The specified columns must exist in the target database and also be part of the Replicat configuration (satisfy the TARGET specification with or without a COLMAP clause).

Only scalar data types are supported by COMPARECOLS as comparison columns. A scalar data type can be used in a WHERE clause, has a single, atomic value and no internal components. Scalar data types supported by Oracle GoldenGate include the following, but not LOBs.

- Numeric data types
- Date data types
- Character data types

Some examples of non-scalar data types are spatial data, user-defined data types, large objects (LOB), XML, reference data types, and RAW. A row being considered for CDR can include non-scalar data so long as the conflict is not in the non-scalar data itself.

To specify conflict resolution routines, use the RESOLVECONFLICT option of MAP. COMPARECOLS and RESOLVECONFLICT can be in any order in the MAP statement.

Getting More Information About Configuring the CDR Feature

See *Administering Oracle GoldenGate* for more information about configuring conflict detection and resolution.

Syntax

```
COMPARECOLS(
{ON UPDATE | ON DELETE}
```

```
{ALL | KEY | KEYINCLUDING (col[,...]) | ALLEXCLUDING (col[,...]) }
[,...]
)
```

{ON UPDATE | ON DELETE}
Specifies whether the before image of the specified columns should be compared for updates or deletes. You can use `ON UPDATE` only, `ON DELETE` only, or both. If using both, specify them within the same `COMPARECOLS` clause. See the example for how to use both.

{ALL | KEY | KEYINCLUDING (*col*[,...]) | ALLEXCLUDING (*col*[,...])}
Specifies the columns for which a before image is captured.

ALL
Compares using all columns in the target table. An error is generated if any corresponding before images are not available in the trail. Using `ALL` imposes the highest processing load for Replicat, but allows conflict-detection comparisons to be performed using all columns for maximum accuracy.

KEY
Compares only the primary key columns. This is the fastest option, but does not permit the most accurate conflict detection, because keys can match but non-key columns could be different.

KEYINCLUDING
Compares the primary key columns and the specified column or columns. This is a reasonable compromise between speed and detection accuracy.

ALLEXCLUDING
Compares all columns except the specified columns. For tables with numerous columns, `ALLEXCLUDING` may be more efficient than `KEYINCLUDING`. Do *not* exclude key columns.

Example 1
In the following example, the key columns plus the `name`, `address`, and `salary` columns are compared for conflicts.

```
MAP src, TARGET tgt
COMPARECOLS (
ON UPDATE KEYINCLUDING (name, address, salary),
ON DELETE KEYINCLUDING (name, address, salary));
```

Example 2
In the following example, the `comment` column is ignored and all other columns are compared for conflicts.

```
MAP src, TARGET tgt
COMPARECOLS (ON UPDATE ALLEXCLUDING (comment))
```

COORDINATED

`COORDINATED` is valid for `MAP`. This option is valid when Replicat is in coordinated mode.

Use the `COORDINATED` option to force transactions made on objects in the same `MAP` statement to be processed as barrier transactions. It causes all of the threads across all `MAP` statements to synchronize to the same trail location. The synchronized position is the beginning of the transaction that contains a record that satisfies a `MAP` that contains the `COORDINATED` keyword. The transaction is then applied

atomically by a single thread, which is either the thread with the lowest thread ID among the currently running threads or a dedicated thread with the ID of 0 if USEDEDICATEDCOORDINATIONTHREAD is specified in the parameter file.

THREAD and THREADRANGE clauses specified in conjunction with COORDINATED are ignored because the record will not be applied by the designated thread(s). The COORDINATED keyword results in temporarily suspending parallelism so that the target tables are in a consistent state before the force-coordinated transaction is applied. After this point, parallel execution commences again.

Replicat by default coordinates transactions in which the primary key is updated, transactions that perform DDL, and certain EVENTACTIONS actions. COORDINATED provides for explicit coordination.

See *Administering Oracle GoldenGate* for more information about Coordinated Replicat.

Syntax

```
COORDINATED
```

Example

The following is an example of the use of the COORDINATED option. In this example, business rules require that the target tables be in a consistent state before Replicat executes transactions that include SQLEXEC operations on the objects specified in the MAP statement. Parallelism must be temporarily converted to serial SQL processing in this case.

Given the following MAP statement, if another thread inserts into t2 a record with a value of 100 for col_val before the insert to t1 is performed by thread 1, then the SQLEXEC will delete the row. If other threads are still processing the record that has the value of 100, the SQLEXEC fails. The results of this MAP statement are, therefore, not predictable.

```
MAP u1.t1, TARGET u2.t1 SQLEXEC (ID test2, QUERY ' delete from u2.t2 where
col_val =100 ', NOPARAMS)), THREAD(1);
```

Conversely, when COORDINATED is used, all of the threads synchronize at a common point, including the one processing the col_val=100 record, thereby removing the ambiguity of the results.

```
MAP u1.t1, TARGET u2.t1 SQLEXEC (ID test2, QUERY ' delete from u2.t2 where
col_val =100 ', NOPARAMS)), THREAD(1), COORDINATED;
```

{DEF| TARGETDEF} ***template***

DEF and TARGETDEF are valid for TABLE and MAP.

Use DEF and TARGETDEF to specify the name of a definitions template that was created by the DEFGEN utility.

- DEF specifies a source-definitions template.
- TARGETDEF specifies a target-definitions template.

A template is based on the definitions of a specific table. It enables new tables that have the same definitions as the original table to be added to the Oracle GoldenGate configuration without running DEFGEN for them, and without having to stop and start

the Oracle GoldenGate process. The definitions in the template are used for definitions lookups.

Getting More Information About Creating Definitions Templates

For more information about templates and DEFGEN, see *Administering Oracle GoldenGate*.

Syntax

```
{DEF | TARGETDEF} template
```

template
The name of one of the following definitions templates generated by the DEFGEN utility:

- Use DEF to specify a source-definitions template generated by the DEF option of the TABLE parameter in the DEFGEN parameter file.
- Use TARGETDEF to specify a target-definitions template generated by the TARGETDEF option of the TABLE parameter in the DEFGEN parameter file.

The definitions contained in the template must be identical to the definitions of the table or tables that are specified in the same TABLE or MAP statement.
Case-sensitivity of the template name is observed when the name is specified the same way that it is stored in the database. Make certain that the template name is specified the same way in both the DEF or TARGETDEF clause in this TABLE or MAP statement, and in the DEFGEN parameter file that created the template.

Example 1
This example shows a case-insensitive template name.

```
MAP acct.cust*, TARGET acct.cust*, DEF custdef;
```

Example 2
This example shows a case-sensitive template name when the database requires quotes to enforce case-sensitivity.

```
TABLE acct.cust*, DEF "CustDef";
```

Example 3
This example shows a case where both DEF and TARGETDEF are used.

```
MAP acct.cust*, TARGET acc.cust*, DEF custdef, TARGETDEF tcustdef;
```

EXCEPTIONSONLY

EXCEPTIONSONLY is valid for MAP.

Use EXCEPTIONSONLY in an exceptions MAP statement intended for error handling. The exceptions MAP statement must follow the MAP statement for which errors are anticipated. The exceptions MAP statement executes only if an error occurs for the last record processed in the preceding regular MAP statement.

To use EXCEPTIONSONLY, use a REPERROR statement with the EXCEPTION option either within the regular MAP statement or at the root of the parameter file. See "REPERROR" for more information.

> **Note:**
>
> If using the Oracle GoldenGate Conflict Detection and Resolution (CDR) feature, a REPERROR with EXCEPTION is not needed. CDR automatically sends all operations that cause errors to the exceptions MAP statement.

The exceptions MAP statement must specify the same source table as in the regular MAP statement, but the target table in the exceptions MAP statement must be an exceptions table.

> **Note:**
>
> See "MAPEXCEPTION (*exceptions_mapping*)" to support wildcarded object names.

Getting More Information About Configuring Exceptions Handling

For more information about configuring exceptions handling with an exceptions MAP statement, see *Administering Oracle GoldenGate*.

Syntax

```
EXCEPTIONSONLY
```

EVENTACTIONS (*action*)

EVENTACTIONS is valid for TABLE and MAP. Some options apply only to one or the other parameter and are noted as such in the descriptions.

Use EVENTACTIONS to cause the process to take a defined action based on a record in the trail, known as the *event record*, that qualifies for a specific filter rule. You can use this system, known as the *event marker system* (or *event marker infrastructure*) to customize processing based on database events. For example, you can suspend a process to perform a transformation or report statistics. The event marker feature is supported for the replication of data changes, but not for initial loads.

To trigger actions that do not require data to be applied to target tables, you can use the Replicat TABLE parameter with filtering options that support EVENTACTIONS. See "TABLE for Replicat" for more information.

> **Caution:**
>
> EVENTACTIONS is not supported if the source database is Teradata and Extract is configured in maximum performance mode.

You may need to combine two or more actions to achieve your goals. When multiple actions are combined, the entire EVENTACTIONS statement is parsed first, and then the specified options execute in order of precedence. The following list shows the order of precedence. The actions listed before Process the record occur before the record is

written to the trail or applied to the target (depending on the process). Actions listed after `Process the record` are executed after the record is processed.

```
TRACE
LOG
CHECKPOINT BEFORE
DISCARD
SHELL
ROLLOVER
```
(Process the record)
```
IGNORE
REPORT
SUSPEND
ABORT
CHECKPOINT AFTER
FORCESTOP
STOP
```

To prevent the event record itself from being processed in the normal manner, use the `IGNORE` or `DISCARD` option. Because `IGNORE` and `DISCARD` are evaluated before the record itself, they prevent the record from being processed. Without those options, `EVENTACTIONS` for Extract writes the record to the trail, and `EVENTACTIONS` for Replicat applies that operation to the target database.

You should take into account the possibility that a transaction could contain two or more records that trigger an event action. In such a case, there could be multiple executions of certain `EVENTACTIONS` specifications. For example, encountering two qualifying records that trigger two successive `ROLLOVER` actions will cause Extract to roll over the trail twice, leaving one of the two files empty of transaction data.

You should also take into account that when the `GETUPDATEBEFORES` parameter is in effect, two records are generated for `UPDATE` operations: a record that contains the before image and a record that contains the after image. An event action is triggered for each of those records when the operation qualifies as an event record. You can use the `BEFOREAFTERINDICATOR` token of the `GGHEADER` column-conversion function as a filter in a `FILTER` clause to qualify the records so that the event action triggers only once, either on the before record or the after record, but not both.

The following example filters on the `BEFORE` indicator. The `EVENTACTION` issues the `ECHO` shell command to output the string 'Triggered on `BEFORE`' to the event log when a `BEFORE` record is encountered.

```
TABLE qasource.test, &
FILTER(@STRFIND('BEFORE', @GETENV('GGHEADER' , 'BEFOREAFTERINDICATOR')) > 0), &
EVENTACTIONS ( shell ('echo --== Triggered on BEFORE ==-- '), LOG);
```

The following shows the result of the event action:

```
013-03-06 17:59:31  INFO    OGG-05301  Shell command output: '--== Triggered
on AFTER ==--'
```

The following example does the same thing, but for the `AFTER` indicator.

```
TABLE qasource.test, &
FILTER(@STRFIND('AFTER', @GETENV('GGHEADER' , 'BEFOREAFTERINDICATOR')) > 0), &
EVENTACTIONS ( shell ('echo --== Triggered on AFTER ==-- '), LOG);
```

In a Teradata configuration where Extract is configured in maximum protection mode, use EVENTACTIONS only in the VAM-sort Extract group. It is not supported by the primary Extract in this configuration because concurrent changes are not sorted in transaction order at this point in the processing stream. For more information, see *Using Oracle GoldenGate for Heterogeneous Databases*.

Getting More Information About Configuring the Event Marker System

See *Administering Oracle GoldenGate* for more information about using EVENTACTIONS and the Event Marker System.

Syntax

```
EVENTACTIONS (
[STOP | SUSPEND | ABORT | FORCESTOP]
[IGNORE [RECORD | TRANSACTION [INCLUDEVENT]]
[DISCARD]
[LOG [INFO | WARNING]]
[REPORT]
[ROLLOVER]
[SHELL 'command' |
  SHELL ('command', VAR variable = {column_name | expression}
  [, ...]) ]
[TRACE[2] file [TRANSACTION] [DDL[INCLUDE] | DDLONLY] [PURGE | APPEND]]
[CHECKPOINT [BEFORE | AFTER | BOTH]]
[, ...]
)
```

STOP
Valid in TABLE for Extract and in MAP for Replicat.
Brings the process to a graceful stop when the specified event record is encountered. The process waits for other operations within event transaction to be completed before stopping. If the transaction is a Replicat grouped or batched transaction, the current group of transactions are applied before the process stops gracefully. The process restarts at the next record after the event record, so long as that record also signified the end of a transaction.
The process logs a message if it cannot stop immediately because a transaction is still open. However, if the event record is encountered within a long-running open transaction, there is no warning message that alerts you to the uncommitted state of the transaction. Therefore, the process may remain running for a long time despite the STOP event.
STOP can be combined with other EVENTACTIONS options except for ABORT and FORCESTOP.

SUSPEND
Valid in TABLE for Extract and in MAP for Replicat.
Pauses the process so that it retains the active context of the current run and can still respond to SEND commands that are issued in GGSCI. When a process is suspended, the INFO command shows it as RUNNING, and the RBA field shows the last checkpoint position.
To resume processing, issue the SEND command with the RESUME option.
To use the CHECKPOINT BEFORE option in conjunction with SUSPEND, the event record must be the start of a transaction for the SUSPEND to take place. That way, if the

process is killed while in the suspended state, the event record with the SUSPEND action is the first record to be reprocessed upon restart. If both CHECKPOINT BERORE and SUSPEND are specified, but the event record is not the start of a transaction, the process abends before SUSPEND can take place.
To use the CHECKPOINT AFTER option in conjunction with SUSPEND, the RESUME command must be issued before the checkpoint can take place, and the event record must be a COMMIT record. If the process is killed while in a SUSPEND state, the process reprocesses the transaction from the last checkpointed position upon restart.
SUSPEND cannot be combined with ABORT but can be combined with all other options.

ABORT
Valid in TABLE for Extract and in MAP for Replicat.
Forces the process to exit immediately when the specified event record is encountered, whether or not there are open transactions. The event record is not processed. A fatal error is written to the log, and the event record is written to the discard file if DISCARD is also specified. The process will undergo recovery on startup.
ABORT can be combined only with CHECKPOINT BEFORE, DISCARD, SHELL, and REPORT.

FORCESTOP
Valid in TABLE for Extract and in MAP for Replicat.
Forces the process to stop gracefully when the specified event record is encountered, but only if the event record is the last operation in the transaction or the only record in the transaction. The record is written normally.
If the event record is encountered within a long-running open transaction, the process writes a warning message to the log and exits immediately, as in ABORT. In this case, recovery may be required on startup. If the FORCESTOP action is triggered in the middle of a long-running transaction, the process exits without a warning message.
FORCESTOP can be combined with other EVENTACTIONS options except for ABORT, STOP, CHECKPOINT AFTER, and CHECKPOINT BOTH. If used with ROLLOVER, the rollover only occurs if the process stops gracefully.

IGNORE [RECORD | TRANSACTION [INCLUDEVENT]]
Valid in TABLE for Extract and in MAP for Replicat.
Ignores some or all of the transaction, depending on the selected action.

- RECORD is the default. It forces the process to ignore only the specified event record, but not the rest of the transaction. No warning or message is written to the log, but the Oracle GoldenGate statistics are updated to show that the record was ignored.
- Use TRANSACTION to ignore the entire transaction that contains the record that triggered the event. If TRANSACTION is used, the event record must be the first one in the transaction. When ignoring a transaction, the event record is also ignored by default. TRANSACTION can be shortened to TRANS.
- Use INCLUDEEVENT with TRANSACTION to propagate the event record to the trail or to the target, but ignore the rest of the associated transaction.

IGNORE can be combined with all other EVENTACTIONS options except ABORT and DISCARD.
An IGNORE action is processed after all the qualification, filtering, mapping, and user-exit operations are processed. The record or transaction is ignored in the final output phase and prevents the record or transaction from being written to the output target (the trail in the case of Extract or the database in the case of Replicat). Therefore, in certain expressions, for example those that include SQLEXEC operations, the SQLEXEC will be executed before the IGNORE is processed. This means that, while the record is

not written to the trail or target database, all of the effects of processing the record through qualification, filtering, mapping and user-exit will occur.
This action is not valid for DDL records. Because DDL operations are autonomous, ignoring a record is equivalent to ignoring the entire transaction.

DISCARD
Valid in TABLE for Extract and in MAP for Replicat.
Causes the process to:

- write the specified event record to the discard file.
- update the Oracle GoldenGate statistics to show that the record was discarded.

The process resumes processing with the next record in the trail.
DISCARD can be combined with all other EVENTACTIONS options except IGNORE.

LOG [INFO | WARNING]
Valid in TABLE for Extract and in MAP for Replicat.
Causes the process to log the event when the specified event record is encountered. The message is written to the report file, to the Oracle GoldenGate error log, and to the system event log.
Use the following options to specify the severity of the message:

- INFO specifies a low-severity informational message. This is the default.
- WARNING specifies a high-severity warning message.

LOG can be combined with all other EVENTACTIONS options except ABORT. If using ABORT, LOG is not needed because ABORT logs a fatal error before the process exits.

REPORT
Valid in TABLE for Extract and in MAP for Replicat.
Causes the process to generate a report file when the specified event record is encountered. This is the same as using the SEND command with the REPORT option in GGSCI.
The REPORT message occurs after the event record is processed (unless DISCARD, IGNORE, or ABORT are used), so the report data will include the event record.
REPORT can be combined with all other EVENTACTIONS options.

ROLLOVER
Valid in TABLE for Extract.
Causes Extract to roll over the trail to a new file when the specified event record is encountered. The ROLLOVER action occurs before Extract writes the event record to the trail file, which causes the record to be the first one in the new file unless DISCARD, IGNORE or ABORT are also used.
ROLLOVER can be combined with all other EVENTACTIONS options except ABORT.
ROLLOVER cannot be combined with ABORT because ROLLOVER does not cause the process to write a checkpoint, and ROLLOVER happens before ABORT.
Without a ROLLOVER checkpoint, ABORT causes Extract to go to its previous checkpoint upon restart, which would be in the previous trail file. In effect, this cancels the rollover.

SHELL '*command*'
Valid in TABLE for Extract and in MAP for Replicat.
Causes the process to execute the specified shell command when the event record is encountered. SHELL '*command*' executes a basic shell command. The command

string is taken at its literal value and sent to the system that way. The command is case-sensitive. Enclose the command string within single quote marks, for example:

```
EVENTACTIONS (SHELL 'echo hello world! > output.txt')
```

If the shell command is successful, the process writes an informational message to the report file and to the event log. Success is based upon the exit status of the command in accordance with the UNIX shell language. In that language, zero indicates success.
If the system call is not successful, the process abends with a fatal error. In the UNIX shell language, non-zero equals failure. Note that the error message relates only to the execution of the `SHELL` command itself, and not the exit status of any subordinate commands. For example, `SHELL` can execute a script successfully, but commands in that script could fail.
`SHELL` can be combined with all other `EVENTACTIONS` options.

SHELL ('*command*', VAR *variable* = {*column_name* | *expression*} [, ...])
Valid in `TABLE` for Extract and in `MAP` for Replicat.
Causes the process to execute the specified shell command when the event record is encountered and supports parameter passing. The command and the parameters are case-sensitive.
When `SHELL` is used with arguments, the entire command and argument strings must be enclosed within parentheses, for example:

```
EVENTACTIONS (SHELL ('Current timestamp: $1  SQLEXEC result is $2 ',VAR $1 =
@GETENV('JULIANTIMESTAMP'),VAR $2 = mytest.description));
```

The input is as follows:

command
Is the command, which is passed literally to the system.

VAR
Is a required keyword that starts the parameter input.

variable
Is the user-defined name of the placeholder variable where the run-time variable value will be substituted. Extra variables that are not used in the command are ignored. Note that any literal in the `SHELL` command that matches a `VAR` variable name is replaced by the substituted `VAR` value. This may have unintended consequences, so test your code before putting it into production.

column_name
Can be the before or after (current) image of a column value.

expression
can be the following, depending on whether column data or DDL is being handled.

- Valid expressions for column data:
  - The value from a `TOKENS` clause in a `TABLE` statement.
  - A return value from any Oracle GoldenGate column-conversion function.
  - A return value from a `SQLEXEC` query or procedure.

- Valid expressions for DDL:
  - Return value from @TOKEN function (Replicat only).
  - Return value from @GETENV function.
  - Return value from other functions that do not reference column data (for example, @DATENOW).
  - Return value from @DDL function.

***TRACE[2] file* [TRANSACTION] [DDL[INCLUDE] | DDLONLY] [PURGE | APPEND]**
Valid in TABLE for Extract and in MAP for Replicat.
Causes process trace information to be written to a trace file when the specified event record is encountered. TRACE provides step-by-step processing information. TRACE2 identifies the code segments on which the process is spending the most time.
By default (without options), standard DML tracing without consideration of transaction boundaries is enabled until the process terminates.

- *file* specifies the name of the trace file and must appear immediately after the TRACE keyword. You can specify a unique trace file, or use the default trace file that is specified with the standalone TRACE or TRACE2 parameter.

  The same trace file can be used across different TABLE or MAP statements in which EVENTACTIONS TRACE is used. If multiple TABLE or MAP statements specify the same trace file name, but the TRACE options are not used consistently, preference is given to the options in the last resolved TABLE or MAP that contains this trace file.
- Use TRANSACTION to enable tracing only until the end of the current transaction, instead of when the process terminates. For Replicat, transaction boundaries are based on the source transaction, not the typical Replicat grouped or batched target transaction. TRANSACTION can be shortened to TRANS. This option is valid only for DML operations.
- DDL[INCLUDE] traces DDL and also DML transactional data processing. Either DDL or DDLINCLUDE is valid.
- DDLONLY traces DDL but does not trace DML transactional data.

  These options are valid only for Replicat. By default DDL tracing is disabled.
- Use PURGE to truncate the trace file before writing additional trace records, or use APPEND to write new trace records at the end of the existing records. APPEND is the default.

TRACE can be combined with all other EVENTACTIONS options except ABORT.
To disable tracing to the specified trace file, issue the GGSCI SEND *process* command with the TRACE OFF *file_name* option.

CHECKPOINT [BEFORE | AFTER | BOTH]
Valid in TABLE for Extract and in MAP for Replicat.
Causes the process to write a checkpoint when the specified event record is encountered. Checkpoint actions provide a context around the processing that is defined in TABLE or MAP statements. This context has a begin point and an end point, thus providing synchronization points for mapping the functions that are performed with SQLEXEC and user exits.

BEFORE
BEFORE for an Extract process writes a checkpoint before Extract writes the event record to the trail. BEFORE for a Replicat process writes a checkpoint before Replicat applies the SQL operation that is contained in the record to the target.
BEFORE requires the event record to be the first record in a transaction. If it is not the first record, the process will abend. Use BEFORE to ensure that all transactions prior to the one that begins with the event record are committed.
When using EVENTACTIONS for a DDL record, note that since each DDL record is autonomous, the DDL record is guaranteed to be the start of a transaction; therefore the CHECKPOINT BEFORE event action is implied for a DDL record.
CHECKPOINT BEFORE can be combined with all EVENTACTIONS options.

AFTER
AFTER for Extract writes a checkpoint after Extract writes the event record to the trail. AFTER for Replicat writes a checkpoint after Replicat applies the SQL operation that is contained in the record to the target.
AFTER flags the checkpoint request as an advisory, meaning that the process will only issue a checkpoint at the next practical opportunity. For example, in the case where the event record is one of a multi-record transaction, the checkpoint will take place at the next transaction boundary, in keeping with the Oracle GoldenGate data-integrity model.
When using EVENTACTIONS for a DDL record, note that since each DDL record is autonomous, the DDL record is guaranteed to be the end (boundary) of a transaction; therefore the CHECKPOINT AFTER event action is implied for a DDL record.
CHECKPOINT AFTER can be combined with all EVENTACTIONS options except ABORT.

BOTH
BOTH combines BEFORE and AFTER. The Extract or Replicat process writes a checkpoint before and after it processes the event record.
CHECKPOINT BOTH can be combined with all EVENTACTIONS options except ABORT.

CHECKPOINT can be shortened to CP.

Example 1
The following example shows how you can configure a process to ignore certain records. When Extract processes any trail record that has name = abc, it ignores the record.

```
TABLE fin.cust, &
WHERE (name = 'abc'), &
EVENTACTIONS (ignore);
```

Example 2
Based on the compatibility and precedence rules of EVENTACTIONS options, DISCARD takes higher precedence than ABORT, so in this example the event record gets written to the discard file before the process abends.

```
MAP fin.cust, TARGET fin.cust2, &
WHERE (name = 'abc'), &
EVENTACTIONS (DISCARD, ABORT);
```

Example 3
The following example executes a SHELL action. It gets the result of a SQLEXEC query and pairs it with the current timestamp.

```
TABLE src.tab &
SQLEXEC (id mytest, query 'select description from lookup &
where pop = :mycol2', params (mycol2 = col2) ), &
EVENTACTIONS (SHELL ('Current timestamp: $1  SQLEXEC result is $2 ', &
VAR $1 = @GETENV('JULIANTIMESTAMP'), VAR $2 = mytest.description));
```

The shell command that results from this example could be similar to the following:

```
'Current timestamp: 212156002704718000  SQLEXEC result is test passed'
```

Example 4
The following example shows how invalid results can occur if a placeholder name conflicts with literal text in the command string. In this example, a placeholder named `$1` is associated with a column value, and the `SHELL` command echoes a literal string that includes `$1`.

```
MAP src.tab1, TARGET targ.tab1 &
EVENTACTIONS (SHELL ('echo Extra charge for $1 is $1', VAR $1 = COL1));
```

This is the unintended result, assuming the column value is `gift wrap`:

```
'Extra charge for gift wrap is gift wrap'
```

Changing the placeholder variable to `$col` results in the correct output:

```
MAP src.tab1, TARGET targ.tab1 &
EVENTACTIONS (SHELL ('echo Extra charge for $col is $1', VAR $col = COL1));
'Extra charge for gift wrap is $1'
```

The following shows similar potential for unintended results:

```
MAP src.tab1, TARGET targ.tab1 &
EVENTACTIONS (SHELL ('Timestamp: $1  Price is $13 > out.txt ', &
VAR $1 = @GETENV('JULIANTIMESTAMP')));
```

The redirected output file might contain a string like this (notice the second timestamp contains an appended value of 3):

```
'Timestamp: 212156002704718000 Price is 2121560027047180003'
```

The intended result is this:

```
'Timestamp: 212156002704718000 Price is $13'
```

Example 5
These examples show different ways to configure tracing.

```
MAP tab1, TARGET tab1 EVENTACTIONS (TRACE ./dirrpt/trace1.txt);
MAP tab2, TARGET tab2 EVENTACTIONS (TRACE ./dirrpt/trace2.txt TRANSACTION);
```

- In the first `MAP` statement, the `trace1.txt` trace file is generated just before the first `tab1` event record is applied to the target. It contains all of the tracing information from that point forward until Replicat terminates or unless tracing is turned off with the GGSCI `SEND REPLICAT` command.
- Because the second `MAP` statement contains the `TRANSACTION` option, the `trace2.txt` file is generated just before the first `tab2` event record is applied to the target, but the tracing stops automatically at the conclusion of the transaction that contains the `tab2` event record.

Example 6
The following shows how EVENTACTIONS with SUSPEND can be used.

- *Case 1*: You are replicating DDL, and you want to ensure that there is enough space in the target database to create a new table. Use EVENTACTIONS with SUSPEND in the MAP statement that maps the CREATE TABLE DDL operation, and then execute a SQL statement in that MAP statement to query the amount of space remaining in a tablespace. If there is enough space, use SEND REPLICAT with RESUME to resume processing immediately; if not, leave Replicat suspended until a DBA can add the space, and then use SEND REPLICAT with RESUME to resume processing.
- *Case 2*: You want to fix unique key violations when they occur on any table. Because Replicat is processing thousands of tables, you do not want to stop the process each time there is a violation, because this would cause Replicat to spend time rebuilding the object cache again upon restart. By using EVENTACTIONS with SUSPEND, you can simply suspend processing until the problem is fixed.
- *Case 3*: At the end of the day, you suspend Replicat to run daily reports, and then resume processing immediately without stopping and restarting the process.

EXITPARAM '*parameter*'

EXITPARAM is valid for TABLE and MAP.

Use EXITPARAM to pass a parameter to the EXIT_PARAMS function of a user exit routine whenever a record from the TABLE or MAP statement is encountered.

Getting More Information about User Exits

See *Administering Oracle GoldenGate* for instructions on how to configure user exits.

See "User Exit Functions" for more information about the syntax for the user exits.

Syntax

```
EXITPARAM 'parameter string'
```

'parameter string'
A parameter that is a literal string. Enclose the parameter within single quotes. You can specify up to 100 characters for the parameter string.

FETCHBEFOREFILTER

FETCHBEFOREFILTER is valid for TABLE.

Use FETCHBEFOREFILTER to fetch columns that are specified with FETCHCOLS or FETCHCOLSEXCEPT before a FILTER operation is executed. Fetching before the filter ensures that values required for the filter are available. Without FETCHBEFOREFILTER, fetches specified with FETCHCOLS or FETCHCOLSEXCEPT are not performed until after filters are executed. Specify FETCHBEFOREFILTER before FILTER in the parameter file.

Do not use this option for tables being processed in pass-through mode by a data-pump Extract group.

Syntax

```
FETCHBEFOREFILTER
```

Example

```
TABLE hr.salary, FETCHCOLS (sal_level),
FETCHBEFOREFILTER,
FILTER (sal_level >= 8)
;
```

{FETCHCOLS | FETCHCOLSEXCEPT} (*column_list*)

FETCHCOLS and FETCHCOLSEXCEPT are valid for TABLE. These options are only valid for the primary Extract.

Use FETCHCOLS and FETCHCOLSEXCEPT to fetch column values from the database when the values are not present in the transaction log record. Use this option if the database only logs the values of columns that were changed in an update operation, but you need to ensure that other column values required for FILTER operations are available.

- FETCHCOLS fetches the specified columns.
- FETCHCOLSEXCEPT fetches all columns except the specified columns. For tables with numerous columns, FETCHCOLSEXCEPT may be more efficient than listing each column with FETCHCOLS.

FETCHCOLS and FETCHCOLSEXCEPT are valid for all databases that are supported by Oracle GoldenGate.

For an Oracle Database, Oracle GoldenGate fetches the values from the undo tablespace through Oracle's Flashback Query mechanism. The query provides a read-consistent image of the columns as of a specific time or SCN. For more information about how Oracle GoldenGate uses Flashback Query.

Instead of using FETCHCOLS or FETCHCOLSEXCEPT, it may be more efficient to enable supplemental logging for the desired columns.

To control fetching and enable a response when a column specified for fetching cannot be located, use the FETCHOPTIONS parameter. To include fetch results in statistical displays generated by the STATS EXTRACT command, use the STATOPTIONS parameter.

If values for columns specified with FETCHCOLS or FETCHCOLSEXCEPT are present in the transaction log, no database fetch is performed. This reduces database overhead.

Syntax

```
{FETCHCOLS | FETCHCOLSEXCEPT} (column [, ...])
```

column
Can be one of the following:

- A column name or a comma-delimited list of column names, as in (col1, col2).
- An asterisk wildcard, as in (*).

Example

The FETCHCOLS clause in this example fetches *only* columns 1 and 3, whereas the FETCHCOLSEXCEPT clause fetches all columns *except* columns 1 and 3.

```
TABLE hq.acct, FETCHCOLS (col1, col3);
TABLE hq.sales, FETCHCOLSEXCEPT (col1, col3);
```

{FETCHMODCOLS | FETCHMODCOLSEXCEPT} (*column_list*)

FETCHMODCOLS and FETCHMODCOLSEXCEPT are valid for TABLE. These options are only valid for the primary Extract.

Use FETCHMODCOLS and FETCHMODCOLSEXCEPT to force column values to be fetched from the database even if the columns are present in the transaction log. These Depending on the database type, a log record can contain all of the columns of a table or only the columns that changed in the given transaction operation.

- FETCHMODCOLS fetches the specified columns.
- FETCHMODCOLSEXCEPT fetches all columns that are present in the transaction log, except the specified columns. For tables with numerous columns, FETCHMODCOLSEXCEPT might be more efficient than listing each column with FETCHMODCOLS.

FETCHMODCOLS and FETCHMODCOLSEXCEPT are valid for all databases that are supported by Oracle GoldenGate.

Observe the following usage guidelines:

- Do not use FETCHMODCOLS and FETCHMODCOLSEXCEPT for key columns.

Syntax

```
{FETCHMODCOLS | FETCHMODCOLSEXCEPT} (column [, ...])
```

(*column* [, ...])
Can be one of the following:

- A column name or a comma-delimited list of column names, as in (col1, col2).
- An asterisk wildcard, as in (*).

Example

The FETCHMODCOLS clause in this example fetches *only* columns 1 and 3, whereas the FETCHMODCOLSEXCEPT clause fetches all columns *except* columns 1 and 3.

```
TABLE hq.acct, FETCHMODCOLS (col1, col3);
TABLE hq.sales, FETCHMODCOLSEXCEPT (col1, col3);
```

FILTER (*filter_clause*)

FILTER is valid for TABLE and MAP.

Use FILTER to select or exclude records based on a numeric value. A filter expression can use conditional operators, Oracle GoldenGate column-conversion functions, or both.

> **Note:**
>
> To filter based on a string, use one of the Oracle GoldenGate string functions. See "Column Conversion Functions" for more information about these functions. You can also use the WHERE option. See "WHERE (*clause*)".

Separate all FILTER components with commas. A FILTER clause can include the following:

- Numbers
- Columns that contain numbers
- Functions that return numbers
- Arithmetic operators:

  + (plus)

  - (minus)

  * (multiply)

  / (divide)

  \ (remainder)
- Comparison operators:

  > (greater than)

  >= (greater than or equal)

  < (less than)

  <= (less than or equal)

  = (equal)

  <> (not equal)

  Results derived from comparisons can be zero (indicating FALSE) or non-zero (indicating TRUE).
- Parentheses (for grouping results in the expression)
- Conjunction operators: AND, OR

Enclose literals in single quotes. Specify case-sensitive column names as they are stored in the database, and enclose them in double quotes if the database requires quotes to enforce case-sensitivity (such as Oracle).

Oracle GoldenGate supports FILTER for columns that have a multi-byte character set.

Getting More Information about Record Filtering

See *Administering Oracle GoldenGate* for more information about FILTER and other filtering options.

Syntax

```
FILTER (
[, ON INSERT | ON UPDATE| ON DELETE]
[, IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE]
, filter_clause
[, RAISEERROR error_number]
)
```

filter_clause
Selects records based on an expression, such as:

```
FILTER ((PRODUCT_PRICE*PRODUCT_AMOUNT) > 10000))
```

You can use the column-conversion functions of Oracle GoldenGate in a filter clause, as in:

```
FILTER (@COMPUTE (PRODUCT_PRICE*PRODUCT_AMOUNT)>10000)
```

Enclose literals in single quotes. Specify case-sensitive column names as they are stored in the database, and enclose them in double quotes if the database requires quotes to enforce case-sensitivity (such as Oracle).
Oracle GoldenGate does not support `FILTER` for columns that have a multi-byte character set or a character set that is incompatible with the character set of the local operating system.
The maximum size of the filter clause is 5,000 bytes.

`ON INSERT | ON UPDATE | ON DELETE`
Restricts record filtering to the specified operation(s). Separate operations with commas, for example:

```
FILTER (ON UPDATE, ON DELETE,
@COMPUTE (PRODUCT_PRICE*PRODUCT_AMOUNT)>10000)
```

The preceding example executes the filter for `UPDATE` and `DELETE` operations, but not `INSERT` operations.

`IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE`
Does not apply the filter for the specified operation(s). Separate operations with commas, for example:

```
FILTER (IGNORE INSERT, @COMPUTE (PRODUCT_PRICE*PRODUCT_AMOUNT)>10000)
```

The preceding example executes the filter on `UPDATE` and `DELETE` operations, but ignores `INSERT` operations.

`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 by Oracle GoldenGate. For example: `RAISEERROR 21000`.

`GETBEFORECOLS (`*`column_specification`*`)`

`GETBEFORECOLS` is valid for `TABLE`.

Use `GETBEFORECOLS` to specify columns for which you want before image to be captured and written to the trail upon an update or delete operation. Use `GETBEFORECOLS` when using the Oracle GoldenGate Conflict Detection and Resolution (CDR) feature in a bi-directional or multi-master configuration. Also use it when using conversion functions or other processing features that require the before image of a record.

For updates, the before image of the specified columns is included in the trail whether or not any given column is modified. In addition to the columns specified in the `GETBEFORECOLS` clause, an Oracle database will also log the before image of other columns that are modified. For other supported databases, you can use the `GETUPDATEBEFORES` parameter to force the inclusion of the before values of other columns that are modified.

> **Note:**
>
> GETUPDATEBEFORES overrides GETBEFORECOLS if both are used in the same parameter file.

To use this parameter, supplemental logging must be enabled for any database that does not log before values by default.

GETBEFORECOLS overrides COMPRESSUPDATES and COMPRESSDELETES if used in the same parameter file.

This parameter is valid for all databases except DB2. For DB2 on all platforms that are supported by Oracle GoldenGate, use the GETUPDATEBEFORES parameter instead of GETBEFORECOLS.

Syntax

```
GETBEFORECOLS(
{ON UPDATE | ON DELETE}
{ALL | KEY | KEYINCLUDING (col[,...])  | KEYANDMOD | | ALLEXCLUDING (col[,...]) }
[,...]
)
```

{ON UPDATE | ON DELETE}
Specifies whether the before image of the specified columns should be captured for updates or deletes. You can use ON UPDATE only, ON DELETE only, or both. If using both, specify them within the same GETBEFORECOLS clause. See the example for how to use both.

{ALL | KEY | KEYINCLUDING (*col*[,...]) | KEYANDMOD | ALLEXCLUDING (*col*[,...])}
Specifies the columns for which a before image is captured.

ALL
Captures a before image of all supported data type columns in the target table, including the primary key; all unsupported columns are skipped and logged in the Extract or Replicat parameter file as an information message. This imposes the highest processing load for Extract, but allows conflict-detection comparisons to be performed using all columns for maximum accuracy.

KEY
Capture before image only for the primary key. This is the fastest option, but does not permit the most accurate conflict detection, because keys can match but non-key columns could be different. KEY is the default.

KEYINCLUDING
Capture before image of the primary key and also the specified column or columns. This is a reasonable compromise between speed and detection accuracy.

KEYANDMOD
Use this option as an extension of the key option for both Extract and Replicat. For update DMLs on the source, Extract logs the key and modified columns.

Replicat on the target will use the KEY and MODIFIED columns during conflict detection in a WHERE clause. With Oracle databases, the modified column is always used for conflict detection by default and this parameter makes it explicit.

ALLEXCLUDING
Capture before image of all columns except the specified columns. For tables with numerous columns, ALLEXCLUDING may be more efficient than KEYINCLUDING. Do *not* exclude key columns.

Example

In the following example, the before images for the key column(s) plus the name, address, and salary are always written to the trail file on update and delete operations.

```
TABLE src,
GETBEFORECOLS (
ON UPDATE KEYINCLUDING (name, address,  salary),
ON DELETE KEYINCLUDING (name, address, salary));
```

HANDLECOLLISIONS | NOHANDLECOLLISIONS

HANDLECOLLISIONS and NOHANDLECOLLISIONS are valid for MAP.

Use HANDLECOLLISIONS and NOHANDLECOLLISIONS to control whether or not Oracle GoldenGate reconciles the results of an initial load with replicated transactional changes that are made to the same tables. When Oracle GoldenGate applies replicated changes after the load is finished, HANDLECOLLISIONS causes Replicat to overwrite duplicate records in the target tables and provides alternate handling of errors for missing records.

HANDLECOLLISIONS and NOHANDLECOLLISIONS can be used globally for all MAP statements in the parameter file or as an ON/OFF switch for groups of tables specified with MAP statements, and they can be used within a MAP statement. When used in a MAP statement, they override the global specifications.

See "HANDLECOLLISIONS | NOHANDLECOLLISIONS" for syntax and usage.

INSERTALLRECORDS

INSERTALLRECORDS is valid for MAP.

Use the INSERTALLRECORDS parameter to convert all mapped operations to INSERT operations on the target. INSERTALLRECORDS can be used at the root level of the parameter file, within a MAP statement, and within a MAPEXCEPTION clause of a MAP statement.

See "INSERTALLRECORDS" for syntax and usage.

INSERTAPPEND | NOINSERTAPPEND

INSERTAPPEND is valid for MAP.

Use the INSERTAPPEND and NOINSERTAPPEND parameters to control whether or not Replicat uses an APPEND hint when it applies INSERT operations to Oracle target tables. These parameters are valid only for Oracle databases.

See "INSERTAPPEND | NOINSERTAPPEND" for syntax and usage.

KEYCOLS (*columns*)

KEYCOLS is valid for TABLE and MAP.

Use KEYCOLS to define one or more columns of the target table as unique. The primary use for KEYCOLS is to define a substitute primary key when a primary key or an appropriate unique index is not available for the table. You can also use KEYCOLS to specify additional columns to use in the row identifier that Replicat uses. Without the availability of a key or KEYCOLS clause, Replicat uses all columns of the table to build its WHERE clause, essentially performing a full table scan.

The columns of a key rendered by KEYCOLS must uniquely identify a row, and they must match the columns that are used as a key on the source table. The source table must contain at least as many key or index columns as the KEYCOLS key specified for the target table. Otherwise, in the event of an update to the source key or index columns, Replicat will not have the before images for the extra target KEYCOL columns.

When defining a substitute key with KEYCOLS, observe the following guidelines:

- If the source and target tables both lack keys or unique indexes, use a KEYCOLS clause in the TABLE parameter and in the MAP parameter, and specify matching sets of columns in each KEYCOLS clause.
- If either of the tables lacks a key or unique index, use KEYCOLS for that table. Specify columns that match the actual key or index columns of the other table. If a matching set cannot be defined with KEYCOLS, you must use KEYCOLS for the source table (TABLE parameter) and for the target table (MAP parameter). Specify matching sets of columns that contain unique values. KEYCOLS overrides a key or unique index.
- If the target table has a larger key than the source table does (or if it has more unique-index columns), use KEYCOLS in the TABLE statement to specify the source columns that match the extra target columns. You must also include the actual source key or index columns in this KEYCOLS clause. Using KEYCOLS in this way ensures that before images are available to Replicat in case the non-key columns are updated on the source.

When using KEYCOLS, make certain that the specified columns are configured for logging so that they are available to Replicat in the trail records. For an Oracle database, you can enable the logging by using the COLS option of the ADD TRANDATA command.

On the target tables, create a unique index on the KEYCOLS-defined key columns. An index improves the speed with which Oracle GoldenGate locates the target rows that it needs to process.

Do not use KEYCOLS for tables being processed in pass-through mode by a data-pump Extract group.

Additional Considerations for KEYCOLS when using Parallel Replicat or Integrated Replicat:

- When using KEYCOLS with ALLOWDUPTARGETMAP, the key columns must be the same for each mapped table. For example, if you map HR.EMP to HR.EMP_TARGET and HR.EMP_BACKUP and if you specify KEYCOLS, they must be the same for both HR.EMP_TARGET and HR.EMP_BACKUP.

- When using KEYCOLS to map from multiple source tables to the same target table, the MAP statements must use the same set of KEYCOLS.

Syntax

```
KEYCOLS (column [, ... ])
```

column
Defines a column to be used as a substitute primary key. If a primary or unique key exists, those columns must be included in the KEYCOLS specification. To specify multiple columns, create a comma-delimited list as in:

```
KEYCOLS (id, name)
```

The following column-types are **not** supported in KEYCOLS:

- Oracle column types **not** supported by KEYCOLS:

  Virtual columns, UDTs, function-based columns, and any columns that are explicitly excluded from the Oracle GoldenGate configuration.

- SQL Server, DB2 LUW, DB2 z/OS, MySQL, and Teradata:

  Columns that contain a timestamp or non-materialized computed column, and any columns excluded from the Oracle GoldenGate configuration. For SQL Server Oracle GoldenGate enforces the total length of data in rows for target tables without a primary key to be below 8000 bytes.

Example

```
TABLE hr.emp, KEYCOLS (id, first, last, birthdate);
```

MAPEXCEPTION (*exceptions_mapping*)

MAPEXCEPTIONS is valid for MAP.

Use MAPEXCEPTION as part of an exceptions MAP statement intended for error handling. MAPEXCEPTION maps failed operations that are flagged as exceptions by the REPERROR parameter to an *exceptions table*. Replicat writes the values of these operations along with other information to the exceptions table.

You can use MAPEXCEPTION within the same MAP statement that includes the source-target table mapping and other standard MAP options. The source and target table names can include wildcards.

When using MAPEXCEPTION, use a REPERROR statement with the EXCEPTION option either within the same MAP statement or at the root of the Replicat parameter file. See "EXCEPTIONSONLY" and "REPERROR".

Getting More Information About Exceptions Handling

For more information about configuring exceptions handling with an exceptions MAP statement, see *Administering Oracle GoldenGate*.

Syntax

```
MAPEXCEPTION (TARGET exceptions_table, INSERTALLRECORDS [,
exception_MAP_options])
```

TARGET *exceptions_table*
The fully qualified name of the exceptions table. Standard Oracle GoldenGate rules for object names apply to the name of the exceptions table. See *Administering Oracle GoldenGate*.

exception_MAP_options
Any valid options of the MAP parameter that you want to apply to the exceptions handling.

INSERTALLRECORDS
Applies all exceptions to the exceptions table as INSERT operations. This parameter is required when using MAPEXCEPTION.

Example

This is an example of how to use MAPEXCEPTION for exceptions mapping. The MAP and TARGET clauses contain wildcard source and target table names. Exceptions that occur when processing any table with a name beginning with TRX will be captured to the fin.trxexceptions table using the specified mapping.

```
MAP src.trx*, TARGET trg.*,
MAPEXCEPTION (TARGET fin.trxexceptions,
INSERTALLRECORDS,
COLMAP (USEDEFAULTS,
ACCT_NO = ACCT_NO,
OPTYPE = @GETENV ('LASTERR', 'OPTYPE'),
DBERR = @GETENV ('LASTERR', 'DBERRNUM'),
DBERRMSG = @GETENV ('LASTERR', 'DBERRMSG')
)
);
```

MAPALLCOLUMNS | NOMAPALLCOLUMNS

MAPALLCOLUMNS and NOMAPALLCOLUMNS are valid for MAP.

Use MAPALLCOLUMNS to obtain unmapped columns (non-key). When this option is specified, Extract or Replicat checks if all source columns are directly mapped to the target without the column mapping function. If any source columns isn't mapped, then the Extract and/or Replicat abends.

See "MAPALLCOLUMNS| NOMAPALLCOLUMNS

MAPINVISIBLECOLUMNS | NOMAPINVISIBLECOLUMNS

MAPINVISIBLECOLUMNS and NOMAPINVISIBLECOLUMNS are valid for MAP.

Use MAPINVISIBLECOLUMNS and NOMAPINVISIBLECOLUMNS to control whether or not Replicat includes invisible columns in Oracle target tables for default column mapping. For invisible columns in Oracle target tables that use explicit column mapping, they are always mapped so do not require this option.

MAPINVISIBLECOLUMNS and NOMAPINVISIBLECOLUMNS can be used in two different ways. When specified at a global level, one parameter remains in effect for all subsequent MAP statements, until the other parameter is specified. When used within a MAP statement, they override the global specifications

See "MAPINVISIBLECOLUMNS | NOMAPINVISIBLECOLUMNS" for syntax and usage.

REPERROR (*error*, *response*)

REPERROR is valid for MAP.

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 at the root level of the parameter file. Multiple REPERROR statements can be applied to the same MAP statement to enable automatic, comprehensive management of errors and interruption-free replication processing.

For syntax and descriptions, see "REPERROR".

RESOLVECONFLICT (*conflict_resolution_specification*)

RESOLVECONFLICT is valid for MAP.

Use RESOLVECONFLICT in a bi-directional or multi-master configuration to specify how Replicat handles conflicts on operations made to the tables in the MAP statement.

Multiple resolutions can be specified for the same conflict type and are executed in the order listed in RESOLVECONFLICT. Multiple resolutions are limited to INSERTROWEXISTS and UPDATEROWEXISTS conflicts only.

RESOLVECONFLICT can be used multiple times in a MAP statement to specify different resolutions for different conflict types.

The following are the data types and platforms that are supported by RESOLVECONFLICT.

- RESOLVECONFLICT supports all databases that are supported by Oracle GoldenGate for Windows and UNIX.
- To use RESOLVECONFLICT, the database must reside on a Windows, Linux, or UNIX system (including those running on NonStop OSS).
- CDR supports data types that can be compared with simple SQL and without explicit conversion. See the individual parameter options for details.
- Do not use RESOLVECONFLICT for columns that contain LOBs, abstract data types (ADT), or user-defined types (UDT).

Getting More Information About Configuring Conflict Resolution

See *Administering Oracle GoldenGate* for detailed instructions on configuring bi-directional replication and conflict resolution, including use cases and examples.

Syntax

```
RESOLVECONFLICT (
{INSERTROWEXISTS | UPDATEROWEXISTS | UPDATEROWMISSING |
   DELETEROWEXISTS | DELETEROWMISSING}
( {DEFAULT | resolution_name},
  {USEMAX (resolution_column) | USEMAXEQ (resolution_column) | USEMIN
(resolution_column) | USEMINEQ (resolution_column) | USEDELTA |
     DISCARD | OVERWRITE | IGNORE}
 )
[, COLS (column[,...])]
)
```

INSERTROWEXISTS | UPDATEROWEXISTS | UPDATEROWMISSING | DELETEROWEXISTS | DELETEROWMISSING
The type of conflict that this resolution handles.

INSERTROWEXISTS
An inserted row violates a uniqueness constraint on the target.

UPDATEROWEXISTS
An updated row exists on the target, but one or more columns have a before image in the trail that is different from the current value in the database.

UPDATEROWMISSING
An updated row does not exist in the target.

DELETEROWEXISTS
A deleted row exists in the target, but one or more columns have a before image in the trail that is different from the current value in the database.

DELETEROWMISSING
A deleted row does not exist in the target.

DEFAULT | *resolution_name*

DEFAULT
The default column group. The resolution that is associated with the `DEFAULT` column group is used for all columns that are not in an explicitly named column group. You must define a `DEFAULT` column group.

resolution_name
A name for a specific column group that is linked to a specific resolution type. Supply a name that identifies the resolution type. Valid values are alphanumeric characters. Avoid spaces and special characters, but underscores are permitted, for example:

```
delta_res_method
```

Use either a named resolution or `DEFAULT`, but not both.

USEMAX (*resolution_column*) | USEMAXEQ (*resolution_column*) | USEMIN (*resolution_column*) | USEMINEQ (*resolution_column*) | USEDELTA | DISCARD | OVERWRITE | IGNORE
The conflict-handler logic that is used to resolve the conflict. Valid resolutions are:

USEMAX
If the value of *resolution_column* in the trail record is greater than the value of the column in the database, the appropriate action is performed.

- (`INSERTROWEXISTS` conflict) Apply the trail record, but change the insert to an update to avoid a uniqueness violation, and overwrite the existing values.
- (`UPDATEROWEXISTS` conflict) Apply the trail record as an update.

USEMAXEQ
If the value of *resolution_column* in the trail record is greater than or equal to the value of the column in the database, the appropriate action is performed.

- (INSERTROWEXISTS conflict) Apply the trail record, but change the insert to an update to avoid a uniqueness violation, and overwrite the existing values.
- (UPDATEROWEXISTS conflict) Apply the trail record as an update.

USEMIN
If the value of *resolution_column* in the trail record is less than the value of the column in the database, the appropriate action is performed:

- (INSERTROWEXISTS conflict) Apply the trail record, but change the insert to an update to avoid a uniqueness violation, and overwrite the existing values.
- (UPDATEROWEXISTS conflict) Apply the update from the trail record.

USEMINEQ
If the value of *resolution_column* in the trail record is less than or equal to the value of the column in the database, the appropriate action is performed:

- (INSERTROWEXISTS conflict) Apply the trail record, but change the insert to an update to avoid a uniqueness violation, and overwrite the existing values.
- (UPDATEROWEXISTS conflict) Apply the update from the trail record.

resolution_column
The name of a NOT NULL column that serves as the resolution column. This column must be part of the column group that is associated with this resolution. The value of the resolution column compared to the current value in the target database determines how a resolution should be applied. The after image of the resolution column is used for the comparison, if available; otherwise the before image value is used. Use a column that can be compared through simple SQL:

- NUMERIC
- DATE
- TIMESTAMP
- CHAR/NCHAR
- VARCHAR/ NVARCHAR

To use a latest-timestamp resolution, use a timestamp column as the *resolution_column* and set the timestamp column to the current time when a row is inserted or updated. If possible, define the resolution column with the SYSTIMESTAMP data type, which supports fractional seconds. When comparisons are performed with sub-second granularity, there is little need for tie-breaking conflict handlers that resolve cases where the value of the resolution column is identical in both trail and target. If you ensure that the value of the timestamp column can only increase or only decrease (depending on the resolution), then USEMAX and USEMIN does not lead to data divergence.

> **Note:**
>
> Do not use a primary key column as the resolution column in a USEMAX statement for the UPDATEROWEXISTS conflict. Otherwise, Replicat abends with an error similar to the following:
>
> ```
> 2013-04-04 10:18:38  ERROR   OGG-01922  Missing  RESOLUTION COLUMN
> NAME while mapping to target table "FIN"."ACCT".
> ```

USEDELTA
(UPDATEROWEXISTS conflict only) Add the difference between the before and after values in the trail record to the current value of the column in the target database. If any of the values is NULL, an error is raised. Base USEDELTA on columns that contain NUMERIC data types. USEDELTA is useful in a multi-node configuration when a row is getting simultaneously updated on multiple nodes. It propagates only the difference in the column values to the other nodes, so that all nodes become synchronized.

DISCARD
(Valid for all conflict types) Retain the current value in the target database, and write the data in the trail record to the discard file.
Use DISCARD with caution, because it can lead to data divergence.

OVERWRITE
(Valid for all conflict types except DELETEROWMISSING) Apply the trail record as follows:

- (INSERTROWEXISTS conflict) Apply the trail record but change the insert to an update to avoid a uniqueness violation, and overwrite the existing values.
- (UPDATEROWEXISTS conflict) Apply the update from the trail record.
- (UPDATEROWMISSING conflict) Apply the trail record but convert the missing UPDATE to an INSERT by using the modified columns from the after image and the unmodified columns from the before image. To convert an update to an insert, the before image of all columns of the row must be available in the trail. Use supplemental logging if the database does not log before images by default, and specify ALL for the Extract GETBEFORECOLS parameter.
- (DELETEROWEXISTS conflict) Apply the delete from the trail record, but use only the primary key columns in the WHERE clause.

Use OVERWRITE with caution, because it can lead to data divergence.

IGNORE
(Valid for all conflict types) Retain the current value in the target database, and ignore the trail record: Do not apply to the target table or a discard file.

COLS (*column*[, ...])
A non-default column group. This is a list of columns in the target database (after mapping) that are linked to, and operated upon by, a specific resolution type. If no column group is specified for a conflict, then all columns are affected by the resolution that is specified for the given conflict.
Alternatively, you can specify a DEFAULT column group, which includes all columns that are not listed in another column group. See the DEFAULT option.

You can specify multiple column groups, each with a different resolution. For example, you could use OVERWRITE for col2 and col3, and you could use USEDELTA for col4. No column in any group can be in any other group. Conflicts for columns in different column groups are resolved separately according to the specified resolution, and in the order listed.
Column groups work as follows:

- For INSERTROWEXISTS and UPDATEROWEXISTS conflicts, you can use different column groups to specify more than one of these conflict types and resolutions per table. Conflicts for columns in different column groups are resolved separately, according to the conflict resolution method specified for the column group.
- For UPDATEROWMISSING, DELETEROWEXISTS, and DELETEROWMISSING, you can use only one column group, and all columns of the table must be in this column group (considered the *default* column group).

Examples

The following examples are explained in detail in *Administering Oracle GoldenGate*.

Example 1
This example demonstrates all conflict types with USEMAX, OVERWRITE, DISCARD.

```
MAP fin.src, TARGET fin.tgt,
    COMPARECOLS (ON UPDATE ALL, ON DELETE ALL),
    RESOLVECONFLICT (UPDATEROWEXISTS, (DEFAULT, USEMAX (last_mod_time)),
    RESOLVECONFLICT (INSERTROWEXISTS, (DEFAULT, USEMAX (last_mod_time)),
    RESOLVECONFLICT (DELETEROWEXISTS, (DEFAULT, OVERWRITE)),
    RESOLVECONFLICT (UPDATEROWMISSING, (DEFAULT, OVERWRITE)),
    RESOLVECONFLICT (DELETEROWMISSING, (DEFAULT, DISCARD)),
    );
```

Example 2
This example demonstrates UPDATEROWEXISTS with USEDELTA and USEMAX.

```
MAP fin.src, TARGET fin.tgt,
    COMPARECOLS
    (ON UPDATE KEYINCLUDING (address, phone, salary, last_mod_time),
    ON DELETE KEYINCLUDING (address, phone, salary, last_mod_time)),
    RESOLVECONFLICT (
    UPDATEROWEXISTS,
    (delta_res_method, USEDELTA, COLS (salary)),
    (DEFAULT, USEMAX (last_mod_time)));
```

Example 3
This example demonstrates UPDATEROWEXISTS with USEDELTA, USEMAX, and IGNORE.

```
MAP fin.src, TARGET fin.tgt,
    COMPARECOLS
    (ON UPDATE ALLEXCLUDING (comment)),
    RESOLVECONFLICT (
    UPDATEROWEXISTS,
    (delta_res_method, USEDELTA, COLS (salary, balance)),
    (max_res_method, USEMAX (last_mod_time), COLS (address, last_mod_time)),
    (DEFAULT, IGNORE));
```

SQLEXEC (*SQL_specification*)

SQLEXEC is valid for TABLE and MAP.

Use SQLEXEC to execute a SQL stored procedure or query from within a MAP statement during Oracle GoldenGate processing. SQLEXEC enables Oracle GoldenGate to communicate directly with the database to perform any work that is supported by the database. This work can be part of the synchronization process, such as retrieving values for column conversion, or it can be independent of extracting or replicating data, such as executing a stored procedure that executes an action within the database.

See "SQLEXEC" for syntax and usage.

SQLPREDICATE 'WHERE *where_clause*'

SQLPREDICATE is valid for TABLE.

Use SQLPREDICATE to include a conventional SQL WHERE clause in the SELECT statement that Extract uses when selecting data from a table in preparation for an initial load. SQLPREDICATE forces the records returned by the selection to be ordered by the key values.

SQLPREDICATE is a faster selection method for initial loads than the WHERE or FILTER options. It affects the SQL statement directly and does not require Extract to fetch all records before filtering them.

For Oracle tables, SQLPREDICATE reduces the amount of data that is stored in the undo segment, which can reduce the incidence of snapshot-too-old errors. This is useful when loading very large tables.

By using a SQLPREDICATE clause, you can partition the rows of a large table among two or more parallel Extract processes. This configuration enables you to take advantage of parallel delivery load processing as well.

SQLPREDICATE also enables you to select data based on a timestamp or other criteria to filter the rows that are extracted and loaded to the target table. SQLPREDICATE can be used for ORDER BY clauses or any other type of selection clause.

Make certain that the WHERE clause contains columns that are part of a key or index. Otherwise, Extract performs a full table scan, which reduces the efficiency of the SELECT statement.

SQLPREDICATE is valid for Oracle, DB2 LUW, DB2 on z/OS, SQL Server, and Teradata databases. Do not use SQLPREDICATE for an Extract group that is configured to synchronize transactional changes. It is only appropriate for an initial load Extract, because it re quires a SELECT statement that selects records directly from tables.

Syntax

```
TABLE source_table, SQLPREDICATE 'WHERE where_clause';
```

WHERE
This is a required keyword.

where_clause
A valid SQL WHERE clause that selects records from the source tables.

Example

```
TABLE hr.emp, SQLPREDICATE 'WHERE state = 'CO' and city = 'DENVER''
```

THREAD (*thread_ID*)

THREAD is valid for MAP. This option is valid when Replicat is in coordinated mode.

Use THREAD to specify that all of the object or objects in the same MAP statement are to be processed by the specified Replicat thread. The specified thread handles filtering, manipulation, delivery to the target, error handling, and other work that is configured for those objects. Wildcards can be used in the TARGET clause when THREAD is used.

All tables that have referential dependencies among one another must be mapped in the same thread. For example, if tables scott.cust and scott.ord have a foreign-key relationship, the following is a possible mapping:

```
MAP scott.cust, TARGET scott.cust, THREAD (5);
MAP scott.ord, TARGET scott.ord, THREAD (5);
```

The thread with the lowest thread ID always processes barrier transactions if the THREAD or THREADRANGE option is omitted. Additionally, and work that is not explicitly assigned to a thread is processed through this thread. For example, if there are threads with IDs ranging from 1 to 10, barrier and non-assigned transactions are performed by thread 1.

To process a MAP statement among multiple threads, see THREADRANGE (*thread_range, column_list*). THREAD and THREADRANGE are mutually exclusive options. Do not use them together in the same MAP statement.

For more information about Replicat modes, see Deciding Which Apply Method to Use in *Using Oracle GoldenGate for Oracle Database* and "BATCHSQL".

Syntax

```
THREAD (thread_ID)
```

thread_ID
A numerical identifier for the thread that will process this MAP statement. Valid values are 1 through the value that was specified with the MAXTHREADS option of the ADD REPLICAT command that created this group. You can use the INFO REPLICAT command to verify the maximum number of threads allowed for a Replicat group. When specifying thread IDs, the following must be true:

- The *total number* of threads specified across all MAP statements of a Replicat group cannot exceed the value of MAXTHREADS.
- No single *thread_ID* value in the Replicat group can be higher than the value of MAXTHREADS. For example, if MAXTHREADS is 25, there cannot be a *thread_ID* of 26 or higher.

If MAXTHREADS was not used, the default maximum number of threads is 25.

Examples

The following examples show some ways to use the THREAD option.

Example 1
In this example, thread 1 processes table cust.

```
MAP scott.cust, TARGET scott.cust, THREAD (1);
```

Example 2
In this example, thread 1 processes all of the tables in the scott schema.

```
MAP scott.*, TARGET scott.*, THREAD (1);
```

Example 3
In this example, the orders table is partitioned among two MAP statements through the use of FILTER (*filter_clause*) and the @RANGE function. For more information about @RANGE, see "@RANGE".

```
MAP scott.orders, TARGET scott.orders, FILTER (@RANGE (1, 2, OID)), THREAD (1);
MAP scott.orders, TARGET scott.orders, FILTER (@RANGE (2, 2, OID)), THREAD (2);
```

THREADRANGE (*thread_range*, *column_list*)

THREADRANGE is valid for MAP. This option is valid when Replicat is in coordinated mode.

Use THREADRANGE to specify that the workload of the target table is to be partitioned evenly among a range of Replicat threads, based on the value of a specified column or columns. For example, if the partitioning is based on the value of a column named ID, and the THREADRANGE value is 1-3, then thread 1 processes rows with ID values from 1 through 10, thread 2 processes rows with ID values from 11 through 20, and thread 3 processes rows with ID values from 21 through 30. The partitioning may not be as absolutely even as shown in the preceding example, depending on the initial calculation of the workload, but it is coordinated so that same row is always processed by the same thread. Each specified thread handles filtering, manipulation, error handling, delivery to the target, and other work for its range of rows.

Partitioning a table across a range of threads may improve apply performance for very large tables or tables that frequently incur long-running transactions or heavy volume, but can be used in other cases, as well. You can process more than one table through the same range of threads.

A wildcarded TARGET clause can be used when THREADRANGE is used if the optional column list is omitted. When using a column list, use separate explicit MAP statements for each table that is using the same thread range.

To process a MAP statement with one specific thread, see THREAD (*thread_ID*). THREAD and THREADRANGE are mutually exclusive options. Do not use them together in the same MAP statement.

Do not specify tables that have referential dependencies among one another in a thread range. Use the THREAD option and process all of those tables with the same thread.

Do not use THREADRANGE to partition sequences. If coordination is required, for example when a sequence is part of a SQLEXEC operation, partition the sequence work to one thread with the THREAD option.

The thread with the lowest thread ID always processes barrier transactions if the THREAD or THREADRANGE option is omitted. Additionally, and work that is not explicitly assigned to a thread is processed through this thread. For example, if there are threads with IDs ranging from 1 to 10, barrier and non-assigned transactions are performed by thread 1.

> **Note:**
>
> The columns specified in a list of columns must exist in the trail file. You can control this using KEYCOLS in the Extract to include this column, or by using FETCHCOLS in the Extract for the column, or by ensuring that the column is part of the supplemental log group and then using LOGALLSUPCOLS.

For more information about Replicat modes, see "Deciding Which Apply Method to Use" in *Using Oracle GoldenGate for Oracle Database* and "BATCHSQL".

Syntax

```
THREADRANGE (lowID-highID, [column[, column][, ...]])
```

lowID
The lowest thread identifier of this range. Valid values are 1 through 500.

highID
The highest thread identifier of this range, which must be a higher number than *lowID*. Valid values are *lowID*+1 through 500. The number of threads in the range cannot exceed the value that was specified with the MAXTHREADS option of the ADD REPLICAT command. If MAXTHREADS was not used, the default maximum number of threads is 25.

[*column*[, *column*][, ...]]
Optional. Specifies one or more unique columns on which to base the row partitioning. To specify multiple columns, use a comma-delimited list, such as col1, col2, col3. When this option is omitted, the partitioning among the threads is based by default on the following columns, in the order of preference shown:

- Primary key
- KEYCOLS clause in the same MAP statement
- All of the columns of the table that are supported by Oracle GoldenGate for use as a key.

Example

The following example divides the orders and order_lines tables between the same two threads, based on the value of the OID column.

```
MAP scott.orders, TARGET scott.orders, THREADRANGE (1-2, OID);
MAP scott.order_lines, TARGET scott.order_lines, THREADRANGE (1-2, OID);
```

TOKENS (*token_definition*)

TOKENS is valid for TABLE.

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 way that Oracle GoldenGate delivers data. For example, you can use token data in column maps, stored procedures called by SQLEXEC, 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.

Do not use this option for tables being processed in pass-through mode by a data-pump Extract group.

The character set of token data is not converted. The token must be in the character set of the source database for Extract and in the character set of the target database for Replicat.

Do not use this option for source tables that are encoded as EBCDIC on a z/OS system if the target tables are not EBCDIC.

For more information about using tokens, see *Administering Oracle GoldenGate*.

Syntax

```
TOKENS (token_name = token_data [, ...])
```

token_name
A name of your choice for the token. It can be any number of valid characters and is not case-sensitive. Multi-byte names are not supported.

token_data
Any valid character string of up to 2000 bytes. The data can be either a literal that is enclosed within single quotes (or double quotes if NOUSEANSISQLQUOTES is in use) or the result of an Oracle GoldenGate column-conversion function.

Example

The following creates tokens named TK-OSUSER, TK-GROUP, and TK-HOST and maps them to token data obtained with the @GETENV function.

```
TABLE ora.oratest, TOKENS (
TK-OSUSER = @GETENV ('GGENVIRONMENT' , 'OSUSERNAME'),
TK-GROUP = @GETENV ('GGENVIRONMENT' , 'GROUPNAME')
TK-HOST =  @GETENV ('GGENVIRONMENT' , 'HOSTNAME'));
```

TRIMSPACES | NOTRIMSPACES

TRIMSPACES and NOTRIMSPACES are valid for TABLE and MAP.

Use TRIMSPACES and NOTRIMSPACES at the root level of a parameter file or within a TABLE or MAP statement to control whether or not trailing spaces in a source CHAR column are truncated when applied to a target CHAR or VARCHAR column. The default is TRIMSPACES.

See "TRIMSPACES | NOTRIMSPACES" for syntax and usage.

TRIMVARSPACES | NOTRIMVARSPACES

TRIMVARSPACES and NOTRIMVARSPACES are valid for TABLE and MAP.

Use TRIMVARSPACES and NOTRIMVARSPACES at the root level of a parameter file or within a TABLE or MAP statement to control whether or not trailing spaces in a source VARCHAR column are truncated when applied to a target CHAR or VARCHAR column. The default is NOTRIMVARSPACES.

See "TRIMVARSPACES | NOTRIMVARSPACES" for syntax and usage.

`WHERE (clause)`

`WHERE` is valid for `TABLE` and `MAP`.

Use `WHERE` to select records based on a conditional statement. `WHERE` does not support the following:

- Columns that have a multi-byte character set or a character set that is incompatible with the character set of the local operating system.
- The evaluation of the before image of a primary key column in the conditional statement as part of a primary key update operation.

Enclose literals in single quotes. Specify case-sensitive column names as they are stored in the database, and enclose them in double quotes if the database requires quotes to enforce case-sensitivity (such as Oracle).

Getting More Information about Record Filtering

See *Administering Oracle GoldenGate* for more information about `WHERE` and other filtering options.

Syntax

```
WHERE (clause)
```

clause
Selects records based on a condition, such as:

```
WHERE (branch = 'NY')
```

Table 2-15 shows permissible `WHERE` operators.

Table 2-15 Permissible WHERE Operators

Operator	Example
Column names	`PRODUCT_AMT` `"Product_Amt"`
Numeric values	`-123, 5500.123`
Literal strings enclosed in single quotes	`'AUTO', 'Ca'`
Column tests	`@NULL`, `@PRESENT`, `@ABSENT` (column is null, present or absent in the record). These tests are built into Oracle GoldenGate.
Comparison operators	`=, <>, >, <, >=, <=`
Conjunctive operators	`AND, OR`
Grouping parentheses	Use open and close parentheses for logical grouping of multiple elements.

Example

The following WHERE example returns all records when the AMOUNT column is over 10,000 and does not cause a record to be discarded when AMOUNT is absent.

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

PARTITIONOBJID

Valid for Integrated Extract.

PARTITIONOBJID is used to specify the object IDs of the partitions to be captured for partitioned tables. PARTITIONOBJID is different from ALTID because PARTITIONOBJID applied to Integrated Extract while ALTID applies to Classic Extract.. For an IO table (with or without overflow area), index segment object ID should be used for partition level filtering. In this case, PARTITIONOBJID in the MAP or TABLE statement specifies the index segment object IDs of the partitions to be extracted.

Syntax

```
MAP/TABLE [container.]schema.table PARTITIONOBJID ptn_object_ID [,
ptn_object_ID]
```

The following restrictions apply:

- Wildcarded table names are not allowed for a MAP/TABLE parameter that contains PARTITIONOBJID.
- DDL Capture and replication is not supported when using PARTITIONOBJID.

Syntax for IO table TABLE statement:

```
TABLE [container.]schema.table PARTITIONOBJID index_segment_object_ID [,
index_segment_object_ID]
```

Syntax for IO table MAP statement:

```
MAP [container.]schema.table PARTITIONOBJID index_segment_object_ID [,
index_segment_object_ID]
```

2.155 TABLE for DEFGEN

Valid For

DEFGEN

Description

Use the TABLE parameter in a DEFGEN parameter file to identify a source table or tables for which you want to run the utility.

You can output definitions for objects that are in different containers in an Oracle container database to the same definitions file. All table attributes must be identical, such as case sensitivity, character set, and the use of the full three-part name. For example, you cannot use two-part names (stripped of their container or catalog by the NOCATALOG parameter) and three-part names in the same definitions file.

Default

None

Syntax

```
TABLE [catalog.]owner.table[, DEF template];
```

[*catalog.*]*owner.table*
The Oracle container database if applicable, and the owner and name of the table. This parameter accepts wildcards. Oracle GoldenGate automatically increases the internal storage to track up to 100,000 wildcard entries.
Oracle GoldenGate preserves the case of the table name. Some databases require a name to be within double quotes to enforce case-sensitivity. Other case-sensitive databases do not require double quotes to enforce case-sensitivity, but the names must be specified the way they are stored in the database. See *Administering Oracle GoldenGate* for how to specify object names.

DEF *template*
Creates a definitions template based on the definitions of the specified table. A template enables new tables that have the same definitions as the specified table to be added during an Oracle GoldenGate process run, without the need to run DEFGEN for them first, and without the need to stop and start the Oracle GoldenGate process to update its definitions cache. To use a template that is generated by DEFGEN, specify it with the `DEF` or `TARGETDEF` option of the `TABLE` or `MAP` statement. To retain case-sensitivity, specify the template name the way you would specify any case-sensitive object in the database. This option is not supported for initial loads.

;
Terminates the `TABLE` statement.

Examples

Example 1

```
TABLE fin.account;
```

Example 2

```
TABLE fin.acc*;
```

Example 3

```
TABLE fin."acct1", DEF "acctdefs";
```

2.156 TABLE for Replicat

Valid For

Replicat

Description

Use the `TABLE` parameter in a Replicat parameter file to specify filtering rules that qualify a data record from the trail to be eligible for an event action that is specified with `EVENTACTIONS`.

> **Caution:**
>
> EVENTACTIONS is not supported if the source database is Teradata and Extract is configured in maximum performance mode.

This form of TABLE statement is similar to that of the Replicat MAP statement, except that there is no mapping of the source table in the data record to a target table by means of a TARGET clause. TABLE for Replicat is solely a means of triggering a non-data action to be taken by Replicat when it encounters an event record. If Replicat is in coordinated mode, all actions are processed through the thread with the lowest thread ID.

Because a target table is not supplied, the following apply:

- No options are available to enable Replicat to map table names or columns to a target table, nor are there options to enable Replicat to manipulate data.
- The ASSUMETARGETDEFS parameter cannot be used in the same parameter file as a Replicat TABLE statement, because ASSUMETARGETDEFS requires the names of target tables so that Replicat can query for table definitions. You must create a source-definitions file to provide the definitions of the source tables to Replicat. Transfer this file to the target system and use the SOURCEDEFS parameter in the Replicat parameter file to specify the path name of the file. See *Administering Oracle GoldenGate* for more information about creating source-definitions files.
- The event record itself is not applied to the target database by Replicat. You must specify either IGNORE or DISCARD as one of the EVENTACTIONS options.

See *Administering Oracle GoldenGate* for information about how to specify object names in TABLE statements.

See *Administering Oracle GoldenGate* for more information about how to use EVENTACTIONS and the Event Marker System.

Syntax

See "TABLE | MAP" for descriptions of the following syntax options.

```
TABLE table_spec,
[, SQLEXEC (SQL_specification), BEFOREFILTER]
[, FILTER (filter_clause)]
[, WHERE (where_clause)]
{, EVENTACTIONS ({IGNORE | DISCARD} [action])}
;
```

Example

The following example enables Replicat tracing for an order transaction that contains an insert operation for a specific order number (order_no = 1). The trace information is written to the order_1.trc trace file. The MAP parameter specifies the mapping of the source table to the target table.

```
MAP sales.order, TARGET rpt.order;
TABLE sales.order,
FILTER (@GETENV ('GGHEADER', 'OPTYPE') = 'INSERT' AND @STREQ (order_no, 1), &
EVENTACTIONS (TRACE order_1.trc TRANSACTION);
```

2.157 TABLEEXCLUDE

Valid For

Extract

Description

Use the TABLEEXCLUDE parameter with the TABLE and SEQUENCE parameters to explicitly exclude tables and sequences from a wildcard specification. The positioning of TABLEEXCLUDE in relation to parameters that specify files or trails determines its effect. Parameters that specify trails or files are: EXTFILE, RMTFILE, EXTTRAIL, RMTTRAIL. The parameter works as follows:

- When a TABLEEXCLUDE specification is placed before any TABLE or SEQUENCE parameters, and also before the parameters that specify trails or files, it applies globally to all trails or files, and to all TABLE and SEQUENCE parameters.
- When a TABLEEXCLUDE specification is placed after a parameter that specifies a trail or file, it is effective only for that trail or file and only for the TABLE or SEQUENCE parameters that are associated with it. Multiple trail or file specifications can be made in a parameter file, each followed by a set of TABLE, SEQUENCE, and TABLEEXCLUDE specifications.

TABLEEXCLUDE is evaluated before evaluating the associated TABLE or SEQUENCE parameter. Thus, the order in which they appear does not make a difference.

When using wildcards, be careful not to place them such that all objects are excluded, leaving nothing to capture. For example, the following captures nothing:

```
TABLE cat1.schema*.tab*;
TABLEEXCLUDE cat1.*.*
```

The default for resolving wildcards is WILDCARDRESOLVE DYNAMIC. Therefore, if a table that is excluded with TABLEEXCLUDE is renamed to a name that satisfies a wildcard, the data will be captured. The DYNAMIC setting enables new table names that satisfy a wildcard to be resolved as soon as they are encountered and included in the Oracle GoldenGate configuration immediately. For more information, see WILDCARDRESOLVE.

See also the EXCLUDEWILDCARDOBJECTSONLY parameter.

Default

None

Syntax

```
TABLEEXCLUDE [container. | catalog.]owner.{table | sequence}
```

***container*.**
If the database requires three-part names, specifies the name or wildcard specification of the Oracle container that contains the object to exclude.

owner
Specifies the name or wildcard specification of the owner, such as the schema, of the object to exclude.

table | sequence
The name or wildcard specification of the object to exclude. To specify object names and wildcards correctly, see *Administering Oracle GoldenGate*.

Example

In this example, `test.tab*` specifies that all tables beginning with `tab` in schema `test` are to be excluded from all trail files. Table `fin.acct` is excluded from trail `ee`. Table `fin.sales` is excluded from trail `ff`.

```
TABLEEXCLUDE  test.tab*
    EXTTRAIL ./dirdat/ee
TABLE pdb1.*.*;
TABLEEXCLUDE pdb1.fin.acct
    EXTTRAIL ./dirdat/ff
TABLE pdb2.*.*;
TABLEEXCLUDE pdb2.fin.sales
```

2.158 TARGETDB

Valid For

Replicat

Description

Use the `TARGETDB` parameter for databases or data sets that require a data source name or identifier to be specified explicitly as part of the connection information. This option is required to identify one of the following:

- The target login database for heterogeneous databases.
- The target data source name (DSN) if Replicat uses ODBC to connect to the database.

Tables specified in `MAP` statements that follow `TARGETDB` are assumed to be from the specified data source.

You might need to use the `USERID` or `USERIDALIAS` parameter in the `TARGETDB` parameter statement, depending on the authentication that is required for the data source.

For databases that allow authentication at the operating-system level, you can specify `TARGETDB` without `USERID` or `USERIDALIAS`.

For DB2 LUW, the `TARGETDB` statement must refer to the database by its real name, rather than by any alias.

See USERID | NOUSERID or USERIDALIAS for more information.

See also SOURCEDB to specify a source data source.

Default

None

Syntax

```
TARGETDB data_source[, SESSIONCHARSET character_set]
```

data_source
The name of the database, catalog, or data source name.
For MySQL databases, you can use the format of `TARGETDB` *`database_name@host_name`* to avoid connection issues caused by the incorrect configuration of `localhost` in the local hosts file. If running MySQL on a port other than the default of 3306, you must specify the port number in the connect string: `TARGETDB` *`database_name@host_name:port`*.

`SESSIONCHARSET` ***`character_set`***
Supports MySQL. Sets the database session character set for the process login session. This parameter overrides any `SESSIONCHARSET` that is specified in the `GLOBALS` file.

Examples

Example 1
This example shows `TARGETDB` without the `USERIDALIAS` parameter.

```
TARGETDB mydb
```

Example 2
This example shows `TARGETDB` with the `USERIDALIAS` parameter.

```
TARGETDB mydb, USERIDALIAS tiger2
```

2.159 TARGETDEFS

Valid For

Extract (primary and data pump)

Description

Use the `TARGETDEFS` parameter to specify a target-definitions file. `TARGETDEFS` names a file on the source system or on an intermediary system that contains data definitions of tables and files that exist on the target system.

You can have multiple `TARGETDEFS` statements in the parameter file if more than one target-definitions file is needed for different definitions, for example if each `TARGETDEFS` file holds the definitions for a specific application.

To generate the target-definitions file, use the DEFGEN utility. Transfer the file to the source or intermediary system before starting Extract.

See *Administering Oracle GoldenGate* for more information about using data-definitions files.

Default

None

Syntax

```
TARGETDEFS file
```

file
The relative or fully qualified path name of the target-definitions file.

Examples

Example 1

```
TARGETDEFS C:\repodbc\sales.def
```

Example 2

```
TARGETDEFS /ggs/dirdef/ODBC/tandem_defs
```

2.160 TCPSOURCETIMER | NOTCPSOURCETIMER

Valid For

Extract

Description

Use the TCPSOURCETIMER and NOTCPSOURCETIMER parameters to manage the timestamps of replicated operations for reporting purposes within the Oracle GoldenGate environment.

TCPSOURCETIMER and NOTCPSOURCETIMER are global parameters and apply to all TABLE statements in the Extract parameter file.

Default

```
TCPSOURCETIMER
```

Syntax

```
TCPSOURCETIMER | NOTCPSOURCETIMER
```

TCPSOURCETIMER
Adjusts the timestamp of data records when they are sent to other systems, making it easier to interpret synchronization lag. This is the default.

NOTCPSOURCETIMER
Retains the original timestamp value. Use NOTCPSOURCETIMER when using timestamp-based conflict resolution in a bidirectional configuration and when using a user token that refers to 'GGHEADER', 'COMMITTIMESTAMP' of the @GETENV column-conversion function.

2.161 THREADOPTIONS

Valid For

Extract

Description

Use the THREADOPTIONS parameter to control how a threaded Extract operates.

Stop and restart GGSCI, Manager, and Extract for the change to take effect.

Default

None

Syntax

```
THREADOPTIONS
[INQUEUESIZE n]
[OUTQUEUESIZE n]
[PROCESSTHREADS SELECT thread_spec | PROCESSTHREADS EXCEPT thread_spec]
[STACKSIZE bytes]
```

INQUEUESIZE *n*
Specifies the number of queue entries in the input queue of each producer Extract thread in an Oracle RAC cluster. Higher values produce better performance for large amounts of data. Lower values move data more quickly in environments with very little activity. Valid values are 16 to 65535. The default is 128. The default should be adequate in most cases, but if you need to increase it, 1000 should be sufficient in most types of environments. See also OUTQUEUESIZE.
In addition to INQUEUESIZE and OUTQUEUESIZE, AIX users might obtain better performance by setting the environment variable AIXTHREAD_SCOPE to S (system scope) which specifies the use of multiple CPUs so that processes can run concurrently. To use system scope, add the following to the .profile file of the user who starts the Manager process or else export the variable manually before starting GGSCI.

```
AIXTHREAD_SCOPE=S
export AIXTHREAD_SCOPE
```

OUTQUEUESIZE *n*
Specifies the number of queue entries in the output queue of each producer Extract thread in an Oracle RAC cluster. Valid values are 8 to 65535. The default is 2048. The default should be adequate in most cases.

[PROCESSTHREADS SELECT *thread_spec* | PROCESSTHREADS EXCEPT *thread_spec*]
Specifies the Extract threads to be processed or to be excluded from processing. Valid values are:

- A single thread ID, such as 1
- A range, such as 1-5

Extract threads are mapped to redo threads. Caution: Excluding any of the Extract threads from being processed excludes that data from being synchronized with the target tables.
Primarily for use when Extract is in Archived Log Only mode (ALO).

[STACKSIZE *bytes*]
Specifies the stack size of each producer Extract thread in an Oracle RAC cluster. Valid values are a range of 65536 - 33554432; the default is 1048576.

2.162 TRACE | TRACE2

Valid For

Extract and Replicat

Description

Use the TRACE and TRACE2 parameters to capture Extract or Replicat processing information to help reveal processing bottlenecks. Both support the tracing of DML and DDL.

Tracing also can be turned on and off by using the SEND EXTRACT or SEND REPLICAT command in GGSCI. .

Contact Oracle Support for assistance if the trace reveals significant processing bottlenecks.

Default

No tracing

Syntax

```
TRACE | TRACE2
[, DDL[INCLUDE] | DDLONLY]
[, [FILE] file_name]
[, THREADS (threadID[, threadID][, ...][, thread_range[, thread_range]
[, ...])]
```

TRACE
Provides step-by-step processing information.

TRACE2
Identifies the code segments on which Extract or Replicat is spending the most time.

DDL[INCLUDE] | DDLONLY
(Replicat only) Enables DDL tracing and specifies how DDL tracing is included in the trace report.

DDL[INCLUDE]
Traces DDL and also traces transactional data processing. This is the default. Either DDL or DDLINCLUDE is valid.

DDLONLY
Traces DDL but does not trace transactional data.

[FILE] *file_name*
The relative or fully qualified name of a file to which Oracle GoldenGate logs the trace information. The FILE keyword is optional, but must be used if other parameter options will follow the file name, for example:

```
TRACE FILE file_name DDLINCLUDE
```

If no other options will follow the file name, the FILE keyword can be omitted, for example:

```
TRACE DDLINCLUDE file_name
```

THREADS (*threadID*[, *threadID*][, ...][, *thread_range*[, *thread_range*][, ...])
Enables tracing only for the specified thread or threads of a coordinated Replicat. Tracing is only performed for threads that are active at runtime.

***threadID*[, *threadID*][, ...]**
Specifies a thread ID or a comma-delimited list of threads in the format of `threadID, threadID, threadID`.

[, *thread_range*[, *thread_range*][, ...]
Specifies a range of threads in the form of `threadIDlow-threadIDhigh` or a comma-delimted list of ranges in the format of `threadIDlow-threadIDhigh, threadIDlow-threadIDhigh`.

A combination of these formats is permitted, such as threadID, threadID, `threadIDlow-threadIDhigh`.
If the Replicat is in coordinated mode and `TRACE` is used with a `THREADS` list or range, a trace file is created for each currently active thread. Each file name is appended with its associated thread ID. This method of identifying trace files by thread ID does not apply when `SEND REPLICAT` is issued by `groupname` with `threadID` (as in `SEND REPLICAT fin003 TRACE`...) or when only one thread is specified with `THREADS`.
Contact Oracle Support for assistance if the trace reveals significant processing bottlenecks.

Examples

Example 1
The following traces to a file named `trace.trc`. If this is a coordinated Replicat group, the tracing applies to all active threads.

```
TRACE /home/ggs/dirrpt/trace.trc
```

Example 2
The following enables tracing for only thread 1. In this case, because only one thread is being traced, the trace file will not have a *`threadID`* extension. The file name is `trace.trc`.

```
TRACE THREADS(1) FILE ./dirrpt/trace.trc
```

Example 3
The following enables tracing for threads 1,2, and 3. Assuming all threads are active, the tracing produces files `trace001`, `trace002`, and `trace003`.

```
TRACE THREADS(1-3) FILE ./dirrpt/trace.trc
```

2.163 TRACETABLE | NOTRACETABLE

Valid For

Extract and Replicat

Description

Use the `TRACETABLE` and `NOTRACETABLE` parameters with Oracle databases to identify a trace table that was created with the `ADD TRACETABLE` command. `TRACETABLE` is

required only if the trace table was created with a name other than the default of GGS_TRACE. If a trace table named GGS_TRACE exists in the database, trace table functionality is enabled automatically, and TRACETABLE is not required.

A trace table is not used when Replicat is in integrated mode. TRACETABLE and NOTRACETABLE are ignored in that mode.

The trace table is used for bidirectional synchronization to identify Replicat transactions to Extract.

If used, TRACETABLE must appear in both the Extract and Replicat parameter files.

- In the Replicat parameter file, TRACETABLE causes Replicat to write an operation to the trace table at the beginning of each transaction.
- In the Extract parameter file, TRACETABLE causes Extract to identify as a Replicat transaction any transaction that begins with an operation on the trace table.

NOTRACETABLE prevents Replicat from writing an operation to the trace table, thus preventing Extract from recognizing Replicat transactions.

To control whether Replicat transactions are extracted by Extract or ignored, use the GETREPLICATES and IGNOREREPLICATES parameters. See "GETREPLICATES | IGNOREREPLICATES" for more information.

For instructions on configuring bidirectional synchronization, see the *Administering Oracle GoldenGate*.

Default

```
GGS_TRACE
```

Syntax

```
TRACETABLE [catalog.]owner.table | NOTRACETABLE
```

[catalog.]owner.table
The catalog (if stored in a consolidation database), owner, and name of the trace table.

Examples

Example 1
This example shows a two-part name.

```
TRACETABLE ggs.excl_trans
```

Example 2
This example shows a three-part name.

```
TRACETABLE user.ggs.excl_trans
```

2.164 TRAILBYTEORDER

Valid For

GLOBALS

Description

Use the TRAILBYTEORDER parameter in the GLOBALS file to set the byte format of the metadata in the trails or files created with the EXTFILE, EXTTRAIL, RMTFILE, and RMTTRAIL parameters. By default, Extract always writes the trail metadata in big endian byte order, regardless of the byte order of the source or target machine.

This parameter affects only the metadata of the trail records. It does not affect the column data.

When used in the GLOBALS file, TRAILBYTEORDER affects all of the files or trails in the same Oracle GoldenGate instance. To specify the byte order of a specific trail or file, use the TRAILBYTEORDER option of the associated EXTFILE, RMTFILE, EXTTRAIL, or RMTTRAIL parameter in the Extract parameter file. In cases where Extract writes to multiple trails or files on different platforms, TRAILBYTEORDER in the Extract parameter file enables the correct byte ordering of each one. When TRAILBYTEORDER is used as an Extract parameter, it overrides any TRAILBYTEORDER specification in the GLOBALS file.

TRAILBYTEORDER reduces the overhead of conversion work when the source and target machines both use little endian. In this case, because the default without TRAILBYTEORDER is BIGENDIAN, the conversion work must be performed from little endian to big endian (to write to trail) and then from big endian to little endian to read the trail on the target. TRAILBYTEORDER prevents unnecessary conversions by allowing you to specify the byte order that is used by both the source and target machines (LITTLEENDIAN) as the byte order of the trail.

In the case where the source byte order is big endian and the target is little endian, where some conversion is required, you can decide whether the conversion takes place at the source or at the target. To perform the conversion on the source, set TRAILBYTEORDER to LITTLEENDIAN. The trail is converted to little endian, and no conversion is needed on the target. To perform the conversion on the target, leave the default set to BIGENDIAN. If the target system of the trail is big endian, TRAILBYTEORDER is not needed, because the default is big endian.

Use the NATIVEENDIAN option for a primary Extract or a data pump if the byte order of the source machine is not known, but you want to keep that format and do not want conversion performed on the source. If nothing is specified with TRAILBYTEORDER, a data pump writes the trail using the same byte order as the input trail, which may not be the desired format.

TRAILBYTEORDER is valid for files that have a FORMAT RELEASE version of at least 12.1. For older versions, this parameter is ignored.

Do not use TRAILBYTEORDER when replicating data to a NonStop system. On the NonStop platform, Oracle GoldenGate only supports BIGENDIAN, the default.

To identify the byte order of the metadata in a trail, use the ENV command of the Logdump utility.

Default

```
BIGENDIAN
```

Syntax

```
TRAILBYTEORDER {BIGENDIAN | LITTLEENDIAN | NATIVEENDIAN}
```

BIGENDIAN
Formats the trail metadata in big endian.

LITTLEENDIAN
Formats the trail metadata in little endian.

NATIVEENDIAN
Formats the trail metadata in the default byte order of the local system. Enables you to make certain the output trail is converted to the native format of the source machine.

Example

```
TRAILBYTEORDER LITTLEENDIAN
```

2.165 TRAILCHARSET

Valid For

Replicat

Description

> **Note:**
>
> This parameter has been replaced by the SOURCECHARSET parameter but may still be retained in existing parameter files for backward compatibility.

Use the TRAILCHARSET parameter to supply a character set for the source data if the trail is written by an Extract version that is earlier than 11.2.1.0.0. In the earlier versions, the source character set is not stored in the trail.

When TRAILCHARSET is used, Replicat uses the specified character set as the source character set when converting character-type columns to the target character set. Replicat issues a warning message when it uses the TRAILCHARSET character set.

By default, Replicat performs character set conversion. This feature is controlled by the CHARSETCONVERSION (default) and NOCHARSETCONVERSION parameters. To use TRAILCHARSET, NOCHARSETCONVERSION cannot be used.

Default

Character set of the operating system

Syntax

```
TRAILCHARSET source_charset [, REPLACEBADCHAR];
```

source_charset
The ICU character-set identifier or an Oracle character-set identifier of the source database. For Oracle databases, Oracle GoldenGate converts an Oracle identifier to the corresponding ICU identifier for conversion to the character set that is specified with the NLS_LANG specification in the SETENV parameter in the Replicat parameter file.

REPLACEBADCHAR
Prevents Replicat from abending when a conversion attempt fails. The failed character is replaced with a replacement character for each target character set. The replacement character is pre-defined in each character set.

Examples

Example 1

```
TRAILCHARSET ISO-8859-9;
```

Example 2

```
TRAILCHARSET windows-932, REPLACEBADCHAR;
```

Example 3

```
TRAILCAHRSET EUC-CN;
```

2.166 TRAILCHARSETASCII

Valid For

Extract for DB2 on z/OS; not valid for Extract data pump or Replicat.

Description

Use `TRAILCHARSETASCII` to cause character data to be written to the trail file in the local ASCII code page of the DB2 subsystem from which data is to be captured.

- Specification of this parameter on a single-byte DB2 z/OS subsystem causes character data from non-Unicode tables to be written to the trail file in the installed ASCII single-byte CCSID. Data from EBCDIC tables is converted to this ASCII CCSID.
- Specification of this parameter on a multi-byte DB2 z/OS subsystem causes Extract to process only ASCII and Unicode tables. Extract abends with an error if it encounters EBCDIC tables. Data from ASCII tables is written to the trail file in the installed ASCII mixed CCSID.

Either `TRAILCHARSETASCII` or `TRAILCHARSETEBCDIC` is required if the target is a multi-byte system. To replicate both ASCII and EBCDIC tables to a multi-byte DB2 z/OS target, process each character set with an Extract process for the EBCDIC tables.

Default

Character data is written in the character set of the host table.

Syntax

```
TRAILCHARSETASCII
```

2.167 TRAILCHARSETEBCDIC

Valid For

Extract for DB2 on z/OS; not valid for Extract data pump or Replicat.

Description

Use TRAILCHARSETEBCDIC to cause character data to be written to the trail file in the local EBCDIC code page of the DB2 subsystem from which data is to be captured.

- Specification of this parameter causes all character data to be written to the trail file in the EBCDIC code page of the job in which Extract is running.
- Specification of this parameter on a single-byte DB2 z/OS subsystem causes character data from non-Unicode tables to be written to the trail file in the installed EBCDIC single-byte CCSID. Data from ASCII tables is converted to this EBCDIC CCSID.
- Specification of this parameter on a multi-byte DB2 z/OS subsystem causes Extract to process only EBCDIC and Unicode tables. Extract abends with an error if it encounters ASCII tables. Data from EBCDIC tables is written to the trail file in the installed EBCDIC mixed CCSID.

Either TRAILCHARSETASCII or TRAILCHARSETEBCDIC is required if the target is a multi-byte system. To replicate both ASCII and EBCDIC tables to a multi-byte DB2 z/OS target, process each character set with an Extract process for the EBCDIC tables.

Default

Character data is written in the character set of the host table.

Syntax

```
TRAILCHARSETEBCDIC
```

2.168 TRAIL_SEQLEN_6D | TRAIL_SEQLEN_9D

Valid For

GLOBALS

Description

Use the TRAIL_SEQLEN_6D | TRAIL_SEQLEN_9D parameters to control the number of digits of trail file sequence numbers that are written by an Extract or a data pump Extract. TRAIL_SEQLEN_6D produces a six digit sequence number for trails and TRAIL_SEQLEN_9D produces nine digits.

Default

```
TRAIL_SEQLEN_9D
```

Syntax

```
[TRAIL_SEQLEN_9D | TRAIL_SEQLEN_6D]
```

Example

```
TRAIL_SEQLEN_9D
TRAIL_SEQLEN_6D
```

2.169 TRANLOGOPTIONS

Valid For

Extract

Description

Use the TRANLOGOPTIONS parameter to control the way that Extract interacts with the transaction log or with the API that passes transaction data, depending on the database or capture mode. You can use multiple TRANLOGOPTIONS statements in the same parameter file, or you can specify multiple options within the same TRANLOGOPTIONS statement, if permissible for those options.

Use a given TRANLOGOPTIONS option only for the database or databases for which it is intended.

Default

None

Syntax

```
TRANLOGOPTIONS {
[{ACTIVATIONIDPADLEN | DATABASEIDPADLEN | THREADPADLEN | SEQPADLEN |
RESETLOGSIDPADLEN} width]
[ADGAPPLYCHECKFREQ seconds | cseconds | mseconds]
[ADGCRETRYCOUNT number]
[ADGTIMEOUT seconds]
[ALLOWTABLECOMPRESSION]
[ALTARCHIVEDLOGFORMAT string] [INSTANCE instance] [THREADID id]
[ALTARCHIVELOGDEST [PRIMARY] [INSTANCE instance] path]
[ALTLOGDEST path | REMOTE]
[ARCHIVEDLOGONLY]
[{ASMBUFSIZE size | DBLOGREADERBUFSIZE size}]
[ASMUSER SYS@ASM_instance, ASMPASSWORD password
   [algorithm ENCRYPTKEY {key_name | DEFAULT}]
[ASMUSERALIAS alias [DOMAIN domain]]
[ASYNCTRANSPROCESSING buffer_size]
[BUFSIZE size]
[CHECKPOINTRETENTIONTIME days]
[CHECKTABLELEVELSUPPLOG]
[COMPLETEARCHIVEDLOGONLY | NOCOMPLETEARCHIVEDLOGONLY]
[COMPLETEARCHIVEDLOGTIMEOUT seconds]
[DB2APIRETRY     retry_count]
[DB2ZV11COMPATIBILITYMODE]
[DBLOGREADER]
[DBLOGREADERBUFSIZE size]
[DLFAILOVER_TIMEOUT seconds]
[DISABLESOFTEOFDELAY]
[EXCLUDETAG [tag | NULL] | [EXCLUDETAG +]
[EXCLUDETRANS transaction]
[EXCLUDEUSER user]
[EXCLUDEUSERID Oracle_uid]
[FAILOVERTARGETDESTID n]
[FETCHLOBIFERROR]
[FETCHPARTIALLOB]
```

```
[FETCHINLINESFLOB]
[FETCHPARTIALXML]
[FILTERTABLE table]
[FORCEFETCHLOB]
[GETCTASDML | NOGETCTASDML]
[HANDLEDLFAILOVER]
[IGNOREDATACAPTURECHANGES | NOIGNOREDATACAPTURECHANGES]
[IGNOREDIRECTLOADINSERTS]
[INCLUDEAUX (AUX_specification)]
[INCLUDEREGIONID | INCLUDEREGIONIDWITHOFFSET]
[INTEGRATEDPARAMS (ENABLE_PROCEDURAL_REPLICATION Y] [, ...])]
[LOGRETENTION [ENABLED | SR | DISABLED]
[LOGSOURCE platform, [PATHMAP path]]

[MAXWARNEOF seconds]

[MAXAUTOCMTTRANSSIZE (range, default)]
[MINEFROMACTIVEDG | NOMINEFROMACTIVEDG]
[MINEFROMSNAPSHOTSTBY | NOMINEFROMSNAPSHOTSTBY]
[MININGUSER {/ | user}[, MININGPASSWORD password]
   [algorithm ENCRYPTKEY {key_name | DEFAULT}] [SYSDBA]
[MININGUSERALIAS alias [DOMAIN domain]]
[TRANLOGOPTIONS MIXEDENDIAN [ON|OFF] [NODDLCHANGEWARNING]]
[NOFLUSH]
[NOMANAGECDCCLEANUP | MANAGECDCCLEANUP]
[PATHMAP NFS_mount_point log_path]
[PREPAREFORUPGRADETOIE | NOPREPAREFORUPGRADETOIE]
[PERFORMANCEPROFILE HIGH|MEDIUM]
[PURGEORPHANEDTRANSACTIONS | NOPURGEORPHANEDTRANSACTIONS]
[QUERYTIMEOUT number]
[QUERYRETRYCOUNT number]
[READQUEUESIZE size]
[READTIMEOUT milliseconds]
[REDO_TRANSPORT_LAG_THRESHOLD seconds]
[REQUIRELONGDATACAPTURECHANGES | NOREQUIRELONGDATACAPTURECHANGES]
[SOURCE_OS_TIMEZONE timezone]
[SUPPRESSNOOOPUPDATES]
[TRANSCLEANUPFREQUENCY minutes]

[TRANCOUNT integer]
[TSLOOKUPBEGINLRI | TSLOOKUPENDLRI]
[VALIDATEINLINESFLOB]
[USE_ROOT_CONTAINER_TIMEZONE]
[USENATIVEOBJSUPPORT | NOUSENATIVEOBJSUPPORT]
[USEPREVRESETLOGSID | NOUSEPREVRESETLOGSID]
[VALIDATEINLINESFLOB]
[VAMCOMPATIBILITY {1 | 2 | 3}]
}
[, ...]
```

{ACTIVATIONIDPADLEN | DATABASEIDPADLEN | THREADPADLEN | SEQPADLEN | RESETLOGSIDPADLEN} *width*

Valid for Extract in classic capture mode for Oracle.

Specifies the minimum default padding length when Extract forms the archive log name using the format specifiers %A, %D, %T, %S, and %R in the ALTARCHIVELOGFORMAT parameter. When the corresponding number is smaller than the field width, it is zero-padded on the left. Table 2-16 shows the specifier that relates to each option and the default length.

Option	Specifier	Default padding length
ACTIVATIONIDPADLEN	%A	8
DATABASEIDPADLEN	%D	8
THREADPADLEN	%T	3 on Windows, 4 on other platforms
SEQPADLEN	%S	5 on Windows, 10 on other platforms
RESETLOGSIDPADLEN	%R	10

ADGAPPLYCHECKFREQ *seconds* | *cseconds* | *mseconds*
Valid for Integrated Extract for Oracle.
Specifies the number of seconds, cseconds, or mseconds that Extract waits between each fetch check for the ADG to catch up. A low number improves latency though increases the number of queries of `current_scn from v$database`. The default is 3 seconds; the minimum is 1 and the maximum is 120 seconds.

ADGCRETRYCOUNT *number*
Valid for Integrated Extract for Oracle.
Specifies the number of times that Extract tries before it reports ADG progress or the reason for no progress when waiting for the ADG to catch up. This value is multiplied with `FETCHCHECKFREQ` to determine approximately how often the ADG progress is reported.

ADGTIMEOUT *seconds*
Valid for Extract in classic capture mode for Oracle.
Sets the interval, in seconds, after which Extract times out if `v$database.current_scn` has not moved past the commit SCN associated with the record for which it needs to process. The default is 30 seconds. The minimum is 5 and the maximum is 2147483646. Supports Extract in classic capture mode when capturing in an Oracle Data Guard environment.

ALLOWTABLECOMPRESSION
Valid for DB2 LUW version 9.5 and 9.7.
Enables Oracle GoldenGate to support tables created with row compression, as long as the tables do not contain LOBs. When this parameter is set, LOB columns are not supported, whether or not the table is compressed. To capture from a source where some tables have row compression and some do not, process the compression-enabled tables with one Extract group and the non-compressed tables with another Extract group.

ALTARCHIVEDLOGFORMAT *string* [INSTANCE *instance*] [THREADID *id*]
Valid for Extract in classic capture mode for Oracle.
Specifies a string that overrides the archive log format of the source database.
In an Oracle RAC environment, use the `ALTARCHIVEDLOGFORMAT` parameter on each node. To ensure that Extract can differentiate between the log streams, use the `INSTANCE` or `THREADID` option. The default log format that is queried from the database for one RAC thread is assumed for all of the other threads if Extract cannot find a log format and nothing is specified with `INSTANCE` or `THREADID`.
The `TRANLOGOPTIONS` statement that includes `ALTARCHIVEDLOGFORMAT` cannot contain any other `TRANLOGOPTIONS` options. Use a separate `TRANLOGOPTIONS` statement to specify other options.

string
Accepts the same specifier as Oracle's parameter `LOG_ARCHIVE_FORMAT`. Extract uses the supplied format specifier to derive the log file name. Example:

```
arch_%T.arc
```

INSTANCE *instance*
For use with Oracle RAC. Applies `ALTARCHIVEDLOGFORMAT` to a specific Oracle instance. Extract verifies the supplied input against the database catalog. Example:

```
TRANLOGOPTIONS ALTARCHIVEDLOGFORMAT &
INSTANCE rac1 log_%t_%s_%r.arc
```

THREADID *id*
For use with Oracle RAC. Specifies the thread number of the instance that has the specified log format.
Example:

```
TRANLOGOPTIONS ALTARCHIVEDLOGFORMAT &
THREADID 2 log_%t_%s_%r.arc
```

**ALTARCHIVELOGDEST [PRIMARY] [INSTANCE *instance*]
[THREADID *id*] *path***
Valid for Extract in classic capture mode for Oracle.
Points Extract to the archived or backup Oracle transaction logs when they reside somewhere other than the default location. Extract first checks the specified location and then checks the default location.

path
Specifies the fully qualified path to the archived logs in the alternate directory. This directory must be NFS mounted to the node where Oracle GoldenGate is running. Use that mount point for `ALTARCHIVELOGDEST`.

PRIMARY
Prevents Extract from checking the default log location if it does not find the log in the alternate location. Only the `ALTARCHIVELOGDEST` path is checked. `PRIMARY` is the default for an Extract that is running in Archived Log Only (ALO) mode; otherwise, it is optional.

INSTANCE *instance*
Applies the specified `ALTARCHIVELOGDEST` behavior to a specific Oracle instance. On RAC, if this option is used, you must specify the `ALTARCHIVELOGDEST` parameter on each node.

THREADID *id*
Applies the specified `ALTARCHIVELOGDEST` behavior to a specific thread number.

ALTLOGDEST *path* | *REMOTE*
Valid for MySQL.
Specifies the location of the MySQL log index file. Extract looks for the log files in this location instead of the database default location. `ALTLOGDEST` can be used when the database configuration does not include the full path name to the logs or when there are multiple MySQL installations on the machine. Extract reads the log index file to find the binary log file that it needs to read. When `ALTLOGDEST` is used, Extract assumes that the logs and the index are in the same location.

Supply the full path name to the directory. On Windows, enclose the path within double quotes if the path contains any spaces, such as in the following example.

```
TRANLOGOPTIONS  ALTLOGDEST "C:\Program Files\MySQL\MySQL Server
5.1\log\test.index"
```

On Linux, use this format:

```
TRANLOGOPTIONS ALTLOGDEST "/mnt/rdbms/mysql/data/logs/binlog.index"
```

To capture from a remote server or in case of remote capture, you only need to specify the `REMOTE` option instead of the index file path on the remote server. For remote capture on both Windows and Linux, specify the following in the Extract parameter file (`capture.prm`):
`TRANLOGOPTIONS ALTLOGDEST REMOTE`
For more information on using the `REMOTE` option, see Setting Logging Parameters.

`ARCHIVEDLOGONLY`
Valid for Extract in classic capture mode for SQL Server.
`ARCHIVEDLOGONLY` causes Extract to read from the transaction log backups exclusively. This parameter puts Extract into Archived Log Only mode (ALO) and allows the ability to run the Extract on a different Windows server, other than the database server. ALO mode is incompatible with the `ACTIVESECONDARYTRUNCSTIONPOINT` parameter.
(Oracle) Valid for Extract in classic capture mode.
`ARCHIVEDLOGONLY` causes Extract to read from the archived logs exclusively, without querying or validating the logs from system views such as `v$log` and `v$archived_log`. This parameter puts Extract into Archived Log Only mode (ALO). By default, Extract does not use archived log-only mode even if the database that it connects to is a physical standby database. For more information about archived-log only mode, see Example 2: Capturing from Multiple Sources in Archive-log Only Mode in *Using Oracle GoldenGate for Oracle Database*.

`ASMBUFSIZE` *`size`*
Valid for Extract in classic capture mode for Oracle.
Controls the maximum size, in bytes, of a read operation into the internal buffer that holds the results of each read of the transaction log. Use this option instead of the `DBLOGREADERBUFSIZE` option if the source Oracle version is one that is:

- 11g that is earlier than 11.2.0.2
- any Oracle 11*g* R1 version

These versions do not support the newer API that is available in Oracle versions that are supported by the `DBLOGREADER` option. It is recommended that you use the `DBLOGREADER` option together with the `DBLOGREADERBUFSIZE` option if supported by your Oracle version.
Higher values increase extraction speed but cause Extract to consume more memory. Low values reduce memory usage but increase I/O because Extract must store data that exceeds the cache size to disk.
The following are the valid ranges and default sizes, in bytes:

- Minimum: size of one block in the redo log
- Maximum: 4 MB
- Default: 2 MB (2097152)

The value of the `BUFSIZE` option must always be at least equal to, or greater than, the value of `DBLOGREADERBUFSIZE`.

ASMUSER SYS@*ASM_instance*, ASMPASSWORD *password* [*algorithm*
ENCRYPTKEY {*key_name* | DEFAULT}]
Valid for Extract in classic capture mode for Oracle.
Specifies credentials for logging in to an ASM instance to read the transaction logs. Can be used instead of ASMUSERALIAS if an Oracle GoldenGate credential store is not being used.

SYS@*ASM_instance*
Specifies the ASM instance for the connection string. The user must be SYS.

password
Is the encrypted password that is copied from the ENCRYPT PASSWORD command results.

algorithm
Specifies the encryption algorithm that was used to encrypt the password: AES128, AES192, AES256, or BLOWFISH.

ENCRYPTKEY *key_name*
Specifies the logical name of a user-created encryption key in the ENCKEYS lookup file. Use if ENCRYPT PASSWORD was used with the KEYNAME *key_name* option.

ENCRYPTKEY DEFAULT
Directs Oracle GoldenGate to use a random key. Use if ENCRYPT PASSWORD was used with the KEYNAME DEFAULT option.

> **Note:**
>
> This parameter does not replace the standard USERID parameter. Both are required in an ASM environment. ASMUSER is not needed if using the DBLOGREADER option to read the logs.

ASMUSERALIAS *alias* [DOMAIN *domain*]
Valid for Extract in classic capture mode for Oracle.
Specifies credentials for logging in to an ASM instance to read the transaction logs. Can be used instead of ASMUSER if an Oracle GoldenGate credential store is being used.

alias
Specifies the alias of the login credential that will be used to log into the ASM instance. This credential must exist in the Oracle GoldenGate credential store. If you are not sure what alias to use, you can inspect the content of the credential store by issuing the INFO CREDENTIALSTORE command..

DOMAIN *domain*
Specifies the domain that is assigned to the specified alias in the credential store.

> **Note:**
>
> This parameter does *not* replace the standard `USERIDALIAS` parameter. Both are required in an ASM environment. `ASMUSERALIAS` is not needed if using the `DBLOGREADER` option to read the logs.

`ASYNCTRANSPROCESSING` *`buffer_size`*
Valid for Extract in integrated capture mode for Oracle.
Controls whether integrated capture runs in asynchronous or synchronous processing mode, and controls the buffer size when Extract is in asynchronous mode. The minimum is 1 and the maximum is 1024; the default is 300.

`ASYNCTRANSPROCESSING` *`buffer_size`*
In asynchronous transaction processing mode, there are two threads of control:

- One thread groups logical change records (LCR) into transactions, does object-level filtering, and does partial rollback processing,
- The other thread formats committed transactions, performs any user-specified transformations, and writes to the trail file.

The transaction buffer is the buffer between these two threads and is used to transfer work from one thread to the other. The default transaction buffer size is 300 committed transactions, but is adjusted downward by the Oracle GoldenGate memory manager if its cache memory is close to being exhausted.

`NOASYNCTRANSPROCESSING`
Disables asynchronous processing and causes Extract to operate in synchronous mode. In this mode, one thread performs all capture work.

`BUFSIZE` *`size`*
Valid for DB2 LUW, DB2 z/OS, and Oracle.
Controls the maximum size, in bytes, of the buffers that are allocated to contain the data that is read from the transaction log.

- For an Oracle source where Extract is processing file-based redo, this parameter also controls the maximum size, in bytes, of a read operation into the buffer.
- For an Oracle source where Extract is processing ASM redo, `TRANLOGOPTIONS` with either `ASMBUFSIZE` or `DBLOGREADERBUFSIZE` controls the read size, and in both cases `BUFSIZE` controls the buffer size. This parameter must be equal to, or greater than, the value that is set for `ASMBUFSIZE` or `DBLOGREADERBUFSIZE` (depending on which is in use.)

High values increase capture speed but cause Extract to consume more memory. Low values reduce memory usage but increase I/O because Extract must store data that exceeds the cache size to disk.
The following are the valid ranges and default sizes, in bytes:
Oracle:

- Minimum: 8,192
- Maximum: 10,000,000

The default buffer size is determined by the source of the redo data:

- For file-based redo, the default is 1000KB (1024000).

- For ASM redo, the default is 1000KB (1024000).
- For `DBLOGREADER` redo, the default is 2MB (2097152).
- For Extract in integrated capture mode, the default is 1000KB (1024000).

DB2 LUW:

- Minimum: 8,192
- Maximum: 10,000,000
- Default: 204,800
- The preceding values must be in multiples of the 4096 page size. Extract will truncate to a multiple if a given value does not meet this requirement.

DB2 z/OS

- Minimum: 36KB (36864)
- Maximum: 32MB (33554432)
- Default: 2MB (2097152)
- The preceding values must be in multiples of the 4096 page size. Extract will truncate to a multiple if a given value does not meet this requirement.
- Each Extract uses a fixed 68KB of ECSA on the DB2 z/OS system that the Extract connects to.

`CHECKPOINTRETENTIONTIME` *days*
Valid for Extract in integrated mode only for Oracle.
Controls the number of days that Extract retains checkpoints before they are purged. Partial days can be specified using decimal values. For example, 8.25 specifies 8 days and 6 hours. When the checkpoint of an Extract in integrated capture mode is purged, LogMiner data dictionary information for the archived redo log file that corresponds to the checkpoint is purged, and the `first_scn` value of the capture process is reset to the SCN value corresponding to the first change in the next archived redo log file. The default is seven days and the minimum is 0.00001.

`CHECKTABLELEVELSUPPLOG`
Valid for Extract in classic capture mode for Oracle.
Causes Extract to send a warning to the report file if it encounters a table for which the `ADD TRANDATA` command was not issued to create an Oracle GoldenGate supplemental log group. `CHECKTABLELEVELSUPPLOG` also verifies whether the key columns in any user-defined log groups for the table are the same as, or a superset of, the key columns of the log group that was created with the `ADD TRANDATA` command. Without key columns, Extract may abend or try to fetch the missing column or columns. By default, `CHECKTABLELEVELSUPPLOG` verification is disabled.

`COMPLETEARCHIVEDLOGONLY | NOCOMPLETEARCHIVEDLOGONLY`
Valid for Extract in classic capture mode for Oracle.
Overrides the default Extract processing of archived logs. This parameter applies when copying production (source) archive logs to a secondary database where they will serve as the data source. Some Oracle programs do not build the archive log from the first byte to the last byte in sequential order, but instead may copy the first 500MB, then the last 500MB, and finally the middle 1000MB, for example. If Extract begins reading at the first byte, it will abend when it reaches the break in the byte sequencing. Waiting for the whole file to be written prevents this problem.

Note that Extract starts to read an archive file before it is completely written to disk, but whether or not it starts to capture data before the file is complete depends on whether COMPLETEARCHIVEDLOGONLY or NOCOMPLETEARCHIVEDLOGONLY is used.

COMPLETEARCHIVEDLOGONLY
This is the default in ALO (archived log only) mode. It forces Extract to wait for the archived log to be written to disk completely before starting to process redo data. In regular mode, use it to override the default of NOCOMPLETEARCHIVEDLOGONLY.

NOCOMPLETEARCHIVEDLOGONLY
This is the default in regular mode. Extract starts processing redo data from an archived log immediately when it becomes available, without waiting for it to be written completely to disk. In ALO mode, use it to override the default of COMPLETEARCHIVEDLOGONLY.

COMPLETEARCHIVEDLOGTIMEOUT *seconds*
Valid for Extract in classic capture mode for Oracle.
Controls the number of seconds that Extract waits, when in COMPLETEARCHIVEDLOGONLY mode, to try again if it cannot validate that a redo log is being completely written to disk. Use this option in conjunction with the COMPLETEARCHIVEDLOGONLY option of TRANLOGOPTIONS. This option is disabled by default, and Extract will abend after ten seconds if it cannot validate that the file is being written to disk. This check is performed by reading the block header from the last block and verifying against the expected sequence number to determine if the last block has been written out. For *seconds* use any value greater than 0. The default is 20.

DB2APIRETRY *number of retries*
If Extract receives an error from the DB2 log reading API db2ReadLog(), then for certain errors the API call is retried. Use the DB2APIRETRY to change the number of retries. The default number of retries is set to 3. SQL code for which the API is retried is SQLCODE -30108.

DB2ZV11COMPATIBILITYMODE
Valid for Extract for DB2 z/OS.
When using Oracle GoldenGate to extract from DB2 z/OS version 11 in some compatibility modes, the Extract process may not programmatically determine the actual database version and an OGG-00551 or OGG-00804 error occurs. Use this option in your Extract parameter file to manually set the correct database version.

DBLOGREADER
Valid for Extract in classic capture mode for Oracle.
Causes Extract to use a newer API that is available as of Oracle 11.2.0.2 and later 11*g* R2 versions. This API uses the database server to access the redo and archive logs. DBLOGREADER can be used to mine logs on regular disks and raw disks, and can be used instead of connecting directly to an Oracle ASM instance. The database system must contain the libraries that contain the API modules and must be running. To use this feature, the Extract database user must have SELECT ANY TRANSACTION privilege.
When used, DBLOGREADER enables Extract to use a read size of up to 4 MB in size. This is controlled with the DBLOGREADERBUFSIZE option. The maximum read size when using the default OCI buffer is 28672 bytes. This is controlled by the ASMBUFSIZE option. A larger buffer may improve the performance of Extract when redo rate is high. When using DBLOGREADER with ASM, do not use the ASMUSER or ASMUSERALIAS and ASMPASSWORD options of TRANLOGOPTIONS. The API uses the user and password

specified with the USERID or USERIDALIAS parameter. For more information about using Oracle GoldenGate with ASM, see Mining ASM-stored Logs in Classic Capture Mode in *Using Oracle GoldenGate for Oracle Database*.

> **Note:**
>
> DBLOGREADERalso can be used when the redo and archive logs are on regular disk or on a raw device.

DBLOGREADERBUFSIZE *size*
Valid for Extract in classic capture mode for Oracle.
Controls the maximum size, in bytes, of a read operation into the internal buffer that holds the results of each read of the transaction log in ASM. High values increase capture speed but cause Extract to consume more memory. Low values reduce memory usage but increase I/O because Extract must store data that exceeds the cache size to disk.
Use DBLOGREADERBUFSIZE together with the DBLOGREADER option if the source ASM instance is Oracle 11.2.0.2 and later 11*g* R2 versions. The newer ASM API in those versions provides better performance than the older one. If the Oracle version is not one of those versions, then ASMBUFSIZE must be used.
The following are the valid ranges and default sizes, in bytes:

- Minimum: size of one block in the redo log
- Maximum: 4 MB

  On AIX, the maximum buffer size is 1048576; any attempt to read more than this maximum will result in error.
- Default: 2 MB (2097152)

The default should be sufficient in most cases.
The value of the BUFSIZE option must always be at least equal to, or greater than, the value of DBLOGREADERBUFSIZE.

DLFAILOVER_TIMEOUT *seconds*
Valid for Extract in integrated mode for Oracle.
Provides a configurable timeout in seconds to allow for standby database reinstatement post-role transition. It is used in conjunction with HANDLEDLFAILOVER to allow Integrated Extract to start up immediately after a role transition. At the end of the timeout period, if the standby database is still not available, then Extract will terminate.
The default is 300 seconds. You can also use centiseconds or milliseconds.

DISABLESOFTEOFDELAY
Valid for Extract only in integrated or classic mode for Oracle and DB2 LUW. .
Use DISABLESOFTEOFDELAY in the Extract parameter file to set that the wait time takes effect when the an EOF status is reported with no records to return.

[EXCLUDETAG [*tag* | NULL] | [EXCLUDETAG +]
Valid for Extract in integrated or classic mode for Oracle.
Use EXCLUDETAG *tag* to direct the Extract process to ignore the individual records that are tagged with the specified redo tag. There is no database release limitation for this parameter though not all releases of Oracle Database support tagging. Compare with

older versions, new trail file contains tag tokens, which would not introduce problems for older trail readers.
Use EXCLUDETAG + to direct the Extract process to ignore the individual records that are tagged with any redo tag.
To tag the individual records, use the DBOPTIONS parameter with the SETTAG option in the Replicat parameter file. Use these parameters to prevent cycling (loop-back) of Replicat the individual records in a bidirectional configuration or to filter other transactions from capture. The default SETTAG value is 00. Valid value is any single Oracle Streams tag or a plus sign (+). A tag value can be up to 2000 hexadecimal digits (0-9 A-F) long. The dbms_streams.set_tag operation is supported by EXCLUDETAG. For more information about Streams tags, see Oracle Streams Tags in *Oracle Streams Replication Administrator's Guide*.

EXCLUDETRANS *transaction*
Valid for Integrated Extract for Oracle.
Specifies the transaction name of the Replicat database user or any other user so that those transactions are not captured by Extract. Use for bi-directional processing to prevent data looping between the databases.
Only valid for Sybase and SQL Server, the default transaction name used by Replicat is ggs_repl. For more information about bidirectional synchronization, see Overview of an Active-Active Configuration in *Administering Oracle GoldenGate*.

EXCLUDEUSER *user*
Valie for DB2 LUW, DB2 for z/OS, DB2 for i, and Oracle.
Specifies the name of the Replicat database user, or of any other user, to be used as a filter that identifies transactions that will be subject to the rules of the GETREPLICATES or IGNOREREPLICATES parameter. Typically, this option is used to identify Replicat transactions in a bi-directional or cascading processing configuration, for the purpose of excluding or capturing them. However, it can be used to identify transactions by any other user, such as those of a specific business application.
You can use EXCLUDEUSER and EXCLUDEUSERID in the same parameter file. Do not use wildcards in either parameter.
The user name must be valid. Oracle GoldenGate queries the database to get the associated user ID and maps the numeric identifier back to the user name. For this reason, if the specified user is dropped and recreated while name resolution is set to the default of DYNAMICRESOLUTION, EXCLUDEUSER remains valid. If the same transaction is performed when name resolution is set to NODYNAMICRESOLUTION, EXCLUDEUSER becomes invalid, and Extract must be stopped and then started to make EXCLUDEUSER take effect. See "DYNAMICRESOLUTION | NODYNAMICRESOLUTION" for more information.

- **DB2 z/OS considerations:** In DB2 for z/OS, the user is always the primary authorization ID of the transaction, which is typically that of the original RACF user who logged on, but also could be a different authorization ID if changed by a transaction processor or by DB2 exits.
- **Oracle considerations:** For an Oracle and DB2 LUW database, multiple EXCLUDEUSER statements can be used. All specified users are considered the same as the Replicat user, in the sense that they are subject to the rules of GETREPLICATES or IGNOREREPLICATES. You must include the IGNOREAPPLOPS parameter for EXCLUDEUSER to operate correctly unlike all other supported databases.

EXCLUDEUSERID *Database_uid*
Valid for Extract for Oracle.

Specifies the database user ID (`uid`) of the Replicat database user, or of any other user, to be used as a filter that identifies transactions that will be subject to the rules of the `GETREPLICATES` or `IGNOREREPLICATES` parameter.
Usage is the same as that of `EXCLUDEUSER`.
`Oracle_uid` is a non-negative integer with a maximum value of 2147483638. There are several system views that can be queried to get the user ID. The simplest one is the `ALL_USERS` view. Oracle GoldenGate does not validate the user ID. If the user that is associated with the specified user ID is dropped and recreated, a new user ID is assigned; therefore, `EXCLUDEUSERID` becomes invalid for that user.

`FAILOVERTARGETDESTID` *`n`*
Valid for Extract for Oracle.
Identifies which standby database the Oracle GoldenGate Extract process must remain behind, with regard to not extracting redo data that has not yet been applied to the Oracle Data Guard standby database. To determine the correct value `for_FAILOVERTARGETDESTID`, the `archive_log_destdatabase` initialization parameter is used with *`n`* being the correct archive log destination identifier. The minimum value is 0, the maximum is 32 and the default 0.

`FETCHLOBIFERROR`
Valid for Extract in classic capture mode Oracle.
Overrides the Extract default of abending if LOB capture from the redo log results in an error, such as incomplete data. It forces Extract to fetch the LOB from the database if there is an error when reading it from the redo log.

> **Caution:**
>
> If a value gets deleted before the fetch occurs, Extract writes a null to the trail. If a value gets updated before a fetch, Extract writes the updated value. To prevent these inaccuracies, try to keep Extract latency low. See Tuning the Performance of Oracle GoldenGateguidelines in *Administering Oracle GoldenGate*for tuning process performance. Also, see FETCHOPTIONS for instructions on setting fetch options.

See also the `FORCEFETCHLOB` option.

`FETCHPARTIALLOB`
Valid for Extract in integrated capture mode for Oracle.
Use this option when replicating to a non-Oracle target or in other conditions where the full LOB image is required. It causes Extract to fetch the full LOB object, instead of using the partial change object from the redo record. By default, the database logmining server sends Extract a whole or partial LOB, depending on whether all or part of the source LOB was updated. To ensure the correct snapshot of the LOB, the Oracle Flashback feature must be enabled for the table and Extract must be configured to use it. The Extract `FETCHOPTIONS` parameter controls fetching and must be set to `USESNAPSHOT` (the default in the absence of `NOUSESNAPSHOT`). Without a Flashback snapshot, Extract fetches the LOB from the table, which may be a different image from the point in time when the redo record was generated.

FETCHINLINESFLOB
Use this option with Classic Extract to force fetching on inline SECUREFILE LOBs even when the LOB is not encrypted, not compressed, and not deduplicated. No value is expected after FETCHINLINESFLOB.

FETCHPARTIALXML
Valid for Extract in integrated capture mode Oracle.
Use this option when replicating to a non-Oracle target or in other conditions where the full LOB image is required. It causes Extract to fetch the full XML document, instead of using the partial change image from the redo record. By default, the database logmining server sends Extract a whole or partial XML document, depending on whether all or part of the source XML was updated. To ensure the correct snapshot of the XML, the Oracle Flashback feature must be enabled for the table and Extract must be configured to use it. The Extract FETCHOPTIONS parameter controls fetching and must be set to USESNAPSHOT (the default in the absence of NOUSESNAPSHOT). Without a Flashback snapshot, Extract fetches the XML document from the table, which may be a different image from the point in time when the redo record was generated.

FILTERTABLE *table*
Valid for Extract for MySQL.
Use this option to specify the fully qualified name of the checkpoint table being used by Replicat. Operations on the checkpoint table will be ignored by the local Extract as a means of preventing data from looping back to the source. For information about creating a checkpoint table, see Creating a Checkpoint Table in *Administering Oracle GoldenGate*. To specify object names and wildcards correctly, see Using Wildcards in Database Object Names in *Administering Oracle GoldenGate*.

FORCEFETCHLOB
Valid for Extract in classic and integrated capture modes for Oracle.
Overrides the default behavior of capturing LOB data from the redo log. Causes LOBs to be fetched from the database by default.

> **Caution:**
>
> If a value gets deleted before the fetch occurs, Extract writes a null to the trail. If a value gets updated before a fetch, Extract writes the updated value. To prevent these inaccuracies, try to keep Extract latency low. The Oracle GoldenGate documentation provides guidelines for tuning process performance. Also, see Interactions Between Fetches from a Table and DDL in *Using Oracle GoldenGate for Oracle Database*for instructions on setting fetch options.

GETCTASDML | NOGETCTASDML
Enables Create Table As Select (CTAS) functionality. When GETCTASDML is enabled, CTAS DMLs are sent from LogMiner and replicated on the target. Execution of the CTAS DDL is suppressed on the target. This parameter cannot be enabled while using the DDL metadata trigger. Trail files produced with the CTAS functionality enabled cannot be consumed by a Replicat version lower than 12.1.2.1.0.
Use GETCTASDML to allow CTAS to replay the inserts of the CTAS thus preserving OIDs during replication. This parameter is only supported with Integrated Dictionary and

any downstream Replicat must be 12.1.2.1 or greater to consume the trail otherwise, there may be divergence.

HANDLEDLFAILOVER
Valid for integrated Extract only for Oracle.
Controls whether Extract will throttle its writing of trail data based on the apply progress of the Fast Start Failover standby database. It is intended to keep Extract at a safe point behind any data loss failover. When using this for data loss in a Data Guard configuration with Fast Start Failover (FSFO), after a role transition you must set the `FAILOVERTARGETDESTID` Extract parameter to identify the archive log destination ID to where the standby can be connected.

IGNOREDATACAPTURECHANGES | NOIGNOREDATACAPTURECHANGES
Valid for DB2 LUW
Controls whether or not Extract captures tables for which `DATA CAPTURE CHANGES` is not set. `IGNOREDATACAPTURECHANGES` ignores tables for which `DATA CAPTURE CHANGES` is not set. Use if tables were specified with a wildcard to ensure that processing continues for tables that do have change capture set. A warning is issued to the error log for tables that were skipped. The default is `NOIGNOREDATACAPTURECHANGES`.

IGNOREDIRECTLOADINSERTS
Valid for Extract in classic capture mode for Oracle.
Causes Extract to ignore all Oracle direct-load `INSERTs`. The default behavior (without this parameter) is to capture Oracle direct-load `INSERTs`. This option applies to Oracle logs with log compatibility of Oracle 10g or later.

INCLUDEAUX (*AUX_specification*)
Directs the Oracle GoldenGate `VAMSERV` component to capture only the specified AUX trails when reading the audit trail. This parameter can improve performance when you know that some AUX trails will not contain data that is to be captured and can be ignored. With this parameter, you specify only the AUX trails that are to be captured. *`AUX_specification`* is a number that represents the AUX trails to be captured. To specify multiple AUX trails, use a comma-delimited list. For example, the following statement includes AUX trails BB & CC = 1, 2.

```
TRANLOGOPTIONS INCLUDEAUX (1, 2)
```

To only include MAT or to exclude all AUX trails, place one space between the parentheses, for example:

```
TRANLOGOPTIONS INCLUDEAUX ( )
```

INCLUDEREGIONID | INCLUDEREGIONIDWITHOFFSET
Valid for Extract in either capture mode for Oracle.
These options support the Oracle data type `TIMESTAMP WITH TIME ZONE` specified as `TZR` (which represents the time zone region, such as `US/Pacific`). By default, Extract abends on `TIMESTAMP WITH TIME ZONE` if it includes a time zone region. These options enable you to handle this timestamp based on the target database type. When Extract detects that the source data type is `TIMESTAMP` and there is a region ID mapping token, Replicat applies the timestamp as follows:

- A `TIMESTAMP WITH TIME ZONE` with `TZR` is applied if the target Oracle version supports it.

- A timestamp with a UTC offset is applied to a non-Oracle database, or to an earlier version of Oracle that does not support `TIMESTAMP WITH TIME ZONE` with `TZR`.

  `INCLUDEREGIONID`
  Use when replicating from an Oracle source to an Oracle target of the same version or later. When `INCLUDEREGIONID` is specified, Extract adds a column index and the two-byte `TMZ` value as a time-zone mapping token and outputs it to the trail in the UTC format of `YYYY-MM-DD HH:MI.SS.FFFFFF +00:00`.

  `INCLUDEREGIONIDWITHOFFSET`
  Use when replicating `TIMESTAMP WITH TIME ZONE` as `TZR` from an Oracle source that is v10*g* or later to an Oracle target that is earlier than 10*g*, or from an Oracle source to a target that is not an Oracle database. When `INCLUDEREGIONIDWITHOFFSET` is specified, Extract converts the time zone region value to a time offset that takes Daylight Saving Time into account based on the date and time. The timestamp data is written to the trail in local time in the format of `YYYY-MM-DD HH:MI.SS.FFFFFF TZH:TZM`, where `TZH:TZM` is the region ID converted time offset.

`LOB_CHUNK_SIZE`
Valid for SQL Server CDC Capture method.
`LOB_CHUNK_SIZE`is a new parameter added for the CDC Capture Method to support large objects. If you have huge LOB data sizes, then you can adjust the `LOB_CHUNK_SIZE` from the default of 4000 bytes, to a higher value up to 65535 bytes, so that the fecth size is increades, reducing the trips needed to fetch the entire LOB
Example: `TRANLOGOPTIONS LOB_CHUNK_SIZE 8000`

`INTEGRATEDPARAMS (`*`parameter value`*` [, ...])`
Valid for Extract in integrated capture mode for Oracle Standard or Enterprise Edition 11.2.0.3 or later.
Passes parameters and values to the Oracle Database logmining server when Extract is in integrated capture mode. The input must be in the form of *`parameter value`*, as in:

```
TRANLOGOPTIONS INTEGRATEDPARAMS (downsream_real_time_mine Y)
```

Valid *`parameter`* specifications and their values are the following:

`max_sga_size`
Specifies the amount of SGA memory that is used by the database logmining server. Can be a positive integer in megabytes. The default is 1 GB if `streams_pool_size` is greater than 1 GB; otherwise, it is 75% of `streams_pool_size`.

`parallelism`
Specifies the number of processes supporting the database logmining server. Can be a positive integer. The default is 2.

`downstream_real_time_mine`
Specifies whether or not integrated capture mines a downstream mining database in real-time mode. A value of `Y` specifies real-time capture and requires standby redo logs to be configured at the downstream mining database. A value of `N` specifies capture from archived logs shipped to the downstream mining database. The default is `N`.

enable_procedural_replication
Enables procedural replication at capture. Procedural replication is disabled by default. A value of Y enables procedural replication. Once this option is turned on for an Extract, it remains on. The parameter value can not be toggled back.

LOGRETENTION [ENABLED [DAYS *n*] | SR | DISABLED]
Valid for Extract in classic capture mode for Oracle Enterprise Edition.
Specifies whether or not Oracle Recovery Manager (RMAN) retains the log files that Extract needs for recovery. When you use the REGISTER EXTRACT command, the logs are retained from the time that the command is issued, based on the current database SCN. The logs are retained until manually deleted. This parameter does not enable or disable RMAN within the database itself.
Other information about LOGRETENTION:

- If the Oracle flash recovery storage area is full, RMAN will purge the archive logs even when needed by Extract. This limitation exists so that the requirements of Extract (and other Oracle replication components) do not interfere with the availability of redo to the database.
- The database user that is assigned to Extract and specified with the USERID or USERIDALIAS parameter must have certain privileges, which are the same as those required for the DBLOGIN parameter.
- LOGRETENTION makes use of an underlying (but non-functioning) Oracle Streams Capture process; thus, it requires the database to be the Enterprise Edition of Oracle version 11g or higher. Oracle Standard Edition and Express Edition do not support this feature. The LOGRETENTION feature can operate concurrently with other Streams installations.

> **Note:**
>
> To support RMAN log retention on Oracle RAC, you must download and install the database patch that is provided in BUGFIX 11879974, before you add the Extract groups.

ENABLED [DAYS *n*]
Enables the log-retention feature. This is the default, except when Extract for an Oracle database is in Archived Log Only (ALO) mode. Extract must be registered with the database by using the REGISTER EXTRACT command with the LOGRETENTION option.
By default, ENABLED honors the SCN of the Bounded Recovery checkpoint and retains the logs up to and including that point. This checkpoint represents the log file of the oldest open *non-persisted* transaction. In the unlikely event that a problem with Bounded Recovery affects the persisted data, the logs that are required to reprocess the oldest open transaction must be available.
You can use the DAYS option to retain the logs for a specific number of days, from 1 to 365 days as a whole number. The default for DAYS is 7 days.
To be more conservative, you can use the SR option instead. See "BR" for more information about the Bounded Recovery feature.

SR
Enables the log-retention feature, but retains logs up to and including the SCN of the log that is required for Extract to revert to standard (normal) recovery

mode. In normal mode, Extract needs access to the log that contains the oldest open transaction that it had in memory. Using SR is a conservative measure that retains more logs than would be retained in Bounded Recovery mode (the default), but it ensures data availability in case Bounded Recovery fails. Extract must be registered with the database by using the REGISTER EXTRACT command with the LOGRETENTION option.

DISABLED
Disables the log-retention feature. This is the default setting when Extract for an Oracle source is operating in Archived Log Only (ALO) mode, but you can override this if needed. If you used the REGISTER EXTRACT command to register Extract, use the UNREGISTER EXTRACT command to unregister the associated Extract group from the database after disabling log retention.

LOGSOURCE *platform*, [PATHMAP *path*]
Valid for Extract in classic capture mode for Oracle.
Specifies the operating system and (optionally) the path name when the redo and/or archived logs are stored on a platform other than the one which is hosting the database. When LOGSOURCE is used, put the entire TRANLOGOPTIONS statement on one line. Do not use ampersand (&) line terminators to split it into multiple lines.

platform
Specifies the platform that hosts the redo or archived logs. Valid values are:

- AIX
- HPUX
- LINUX
- MVS
- SOLARIS
- VMS
- WINDOWS
- S390

To maintain correct data alignment, the specified *platform* and the platform that Extract is running on must have the same endian order and bit width (as in 32-bit or 64-bit). The following are compatible endian platforms:

- Big endian: AIX, HPUX, MVS, SOLARIS, S290
- Little endian: LINUX, VMS, WINDOWS

For example when running Extract on HPUX, a LOGSOURCE *platform* setting of AIX is valid but LINUX is not.

PATHMAP *path*
Specifies the path to the logs.

MAXWARNEOF *seconds*
Valid for Extract in classic capture mode for Oracle.
Specifies the number of seconds that Extract waits for a new log file to become available before generating a warning message. Extract generates only one warning message for a given sequence number. If MAXWARNEOF is not specified, Extract waits

for one hour by default. A value of 0 omits the warning no matter how long Extract waits. The minimum value is 0 and the default is 3600.

MAXAUTOCMTTRANSSIZE (*range*, *default*)
Valid for DB2 for i only
Provides the range of the maximum autocommited transaction size.
DB2 for i autocommited records (journal entry has CCID equal to 0) do not have a commit record in the journal and therefore Oracle GoldenGate must create an implicit transaction to include these records in the trail. The default allows for multiple records to be included in a single transaction at the expense of accuracy of the indicated IO Time for each record because the IO time is based on the commit for the transaction. This improves the overall efficiency of handling these records.
This parameter sets the maximum number of records that will be included in an implicitly created transaction, but the number could be less if any other type of entry is seen in the journal before the maximum is reached. This behavior avoids issues with overlap for checkpoints on records that belong to explicitly committed records.
Setting the value for this parameter to 1 will provide an accurate IO time for each record in the trail for records that are autocommitted (have a CCID of 0 in the journal entry), at the potential expense of throughput for the Extract. The value of this parameter also affects the maximum potential size of a cached transaction for these records in memory. Setting it to a lower value causes the transaction memory to be lower if the Extract is able to store the maximum number of entries per implicit transaction. By definition there can only be one such implicit transaction in memory at any given time since any other transaction records will cause an immediate commit to the trail of any records in an implicit transaction already in memory.
The default range is between 1-10000 and the default value is 1000.

MINEFROMACTIVEDG
Valid for Extract in classic capture mode for Oracle.
Specifies that Extract is allowed to mine redo from an Active Data Guard instance. Without this parameter set, Extract will abend with an error. Supports Extract in classic capture mode when capturing in an Oracle Data Guard environment.
`MINEFROMACTIVEDG` does not support `DBLOGREADER`,it only supports `ASMUSER` for reading the redo logs in the ASM storage.

MININGUSER {/ | *user*} [, MININGPASSWORD *password]*
[*algorithm* ENCRYPTKEY {*key_name* | DEFAULT}] [SYSDBA]]
Valid for Extract in integrated capture mode for Oracle.
Specifies login credentials for Extract to log in to a downstream Oracle mining database to interact with the logmining server. Can be used instead of the `MININGUSERALIAS` option if an Oracle GoldenGate credential store is not being used. This user must:

- Have the privileges granted in `dbms_goldengate_auth.grant_admin_privilege`.
- Be the user that issues the `MININGDBLOGIN` or `MININGDBLOGINALIAS` and `REGISTER EXTRACT` or `UNREGISTER EXTRACT` commands for the Extract group that is associated with this `MININGUSERALIAS`.
- Not be changed while Extract is in integrated capture mode.

  /
  Directs Oracle GoldenGate to use an operating-system login for Oracle, not a database user login. Use this argument only if the database allows authentication at the operating-system level. Bypassing database-level authentication eliminates

the need to update Oracle GoldenGate parameter files if application passwords frequently change.
To use this option, the correct user name must exist in the database, in relation to the value of the Oracle OS_AUTHENT_PREFIX initialization parameter. The value set with OS_AUTHENT_PREFIX is concatenated to the beginning of a user's operating system account name and then compared to the database name. Those two names must match.
When OS_AUTHENT_PREFIX is set to ' ' (a null string), the user name must be created with IDENTIFIED EXTERNALLY. For example, if the OS user name is ogg, you would use the following to create the database user:

```
CREATE USER ogg IDENTIFIED EXTERNALLY;
```

When OS_AUTHENT_PREFIX is set to OPS$ or another string, the user name must be created in the format of:

```
OS_AUTHENT_PREFIX_value OS_user_name
```

For example, if the OS user name is ogg, you would use the following to create the database user:

```
CREATE USER ops$ogg IDENTIFIED BY oggpassword;
```

user
Specifies the name of the mining database user or a SQL*Net connect string.

password
The user's password. Use when database authentication is required to specify the password for the database user. If the password was encrypted by means of the ENCRYPT PASSWORD command, supply the encrypted password; otherwise, use the clear-text password. If the password is case-sensitive, type it that way. If either the user ID or password changes, the change must be made in the Oracle GoldenGate parameter files, including the re-encryption of the password if necessary.

algorithm
Specifies the encryption algorithm that was used to encrypt the password with ENCRYPT PASSWORD. Can be one of:

```
AES128
AES192
AES256
BLOWFISH
```

ENCRYPTKEY {*key_name* | DEFAULT}
Specifies the encryption key that was specified with ENCRYPT PASSWORD.

- ENCRYPTKEY *key_name* specifies the logical name of a user-created encryption key in the ENCKEYS lookup file. Use if ENCRYPT PASSWORD was used with the KEYNAME *key_name* option.
- ENCRYPTKEY DEFAULT directs Oracle GoldenGate to use a random key. Use if ENCRYPT PASSWORD was used with the KEYNAME DEFAULT option.

SYSDBA
Specifies that the user logs in as sysdba.

MINEFROMSNAPSHOTSTBY | NOMINEFROMSNAPSHOTSTBY
Valid for Oracle.
Controls whether or not Oracle GoldenGate can capture from redo that is archived by a snapshot standby database. MINEFROMSNAPSHOTSTBY enables Extract to run on a snapshot standby in classic capture mode or in integrated capture mode in an upstream configuration; running in a downstream configuration is not supported because the snapshot standby database does not ship its redo logs to another database.
The default is NOMINEFROMSNAPSHOTSTBY, which prevents Extract from capturing from a database that is a snapshot. Extract cannot run on a physical standby database and will abend if its source snapshot database is converted to a physical database.

MININGUSERALIAS *alias*
Valid for Extract in integrated capture mode for Oracle.
Specifies the alias for the login credentials that Extract uses to log in to a downstream Oracle mining database to interact with the logmining server. Can be used instead of MININGUSER if an Oracle GoldenGate credential store is being used.
This alias must be:

- Associated with a database user login credential that is stored in the local Oracle GoldenGate credential store. This user must have the privileges granted in dbms_goldengate_auth.grant_admin_privilege.
- The user that issues the MININGDBLOGIN or MININGDBLOGINALIAS and REGISTER EXTRACT or UNREGISTER EXTRACT commands for the Extract group that is associated with this MININGUSERALIAS.

This alias and user must not be changed while Extract is in integrated capture mode.

MIXEDENDIAN [ON|OFF]
Valid for DB2 LUW with Oracle GoldenGate primary Extract
Oracle GoldenGate Extract for DB2 LUW supports cross-endian capture where the database and Oracle GoldenGate are running on different byte order servers. Detection of byte order is automatic for DB2 LUW database version 10.5 and higher. If you need to disable auto-detection on DB2 LUW 10.5, then you can override it by specifying this parameter. For version 10.1, the parameter must be used in the Extract parameter file for the cross-endian capture. By default, the value is set to OFF for version 10.1.
Syntax:
TRANLOGOPTIONS MIXEDENDIAN [ON|OFF]
ON: If this is set, then the Extract assumes that the database and Oracle GoldenGate are running on servers with a different byte order and necessary byte reversal conversion is performed.
OFF: If this is set, then the Extract assumes that the database and Oracle GoldenGate are running on servers with the same byte order and no byte order reversal conversion is performed.

NOFLUSH
Valid for DB2 z/OS.
Inhibits the flushing of log buffers.

NOMANAGECDCCLENUP | MANAGECDCCLEANUP
Valid for SQL Server.
When enabling Supplemental Logging for SQL Server tables for the CDC Capture method, data in the CDC staging tables and other CDC system tables need to be purged. Use either the Oracle GoldenGate CDC Cleanup job or the SQL

Server CDC Cleanup job to purge these tables. By default, when Extract starts MANAGECDCCLEANUP is enabled and instructs Extract to check for the existence of the Oracle GoldenGate CDC Cleanup job and this is the recommended approach. By default, Extract requires that the Oracle GoldenGate CDC Cleanup job be created using the ogg_cdc_cleanup_setup.bat program or the Extract cannot start. The NOMANAGECDCCLEANUP option instructs Extract not to check for the existence of the Oracle GoldenGate CDC Cleanup job.

PATHMAP *NFS_mount_point log_path*
Valid for Extract in classic capture mode for Oracle.
Specifies the location of the redo and/or archived logs when they are stored on a system other than the one which is hosting the database. More than one PATHMAP statement can be used. When PATHMAP is used, put the entire TRANLOGOPTIONS statement on one line. Do not use ampersand (&) line terminators to split it into multiple lines.
PATHMAP can be used with the LOGSOURCE option if the system is a different platform from the one that hosts the database.

NFS_mount_point
Specifies the NFS mount point of the remote system where the logs are stored.

log_path
The path to the logs on the remote system. The path must follow the mount point specification.

PREPAREFORUPGRADETOIE | NOPREPAREFORUPGRADETOIE
Valid when upgrading from Classic to Integrated Extract on Oracle RAC for Oracle.
When upgrading on Oracle RAC from Classic to Integrated Extract, you must set the PREPAREFORUPGRADETOIE option before stopping Classic Extract for the upgrade then wait for the information message in the report file that indicates that the parameter has taken effect before proceeding with the upgrade. For detailed upgrade instructions, see *Upgrading Oracle GoldenGate*.

PREPAREFORUPGRADETOIE
Set PREPAREFORUPGRADETOIE in the Extract parameter file, which requires a restart of Extract, or you can set it dynamically for a running extract from GGSCI using this command:
SEND EXTRACT *extract_name* TRANLOGOPTIONS PREPAREFORUPGRADETOIE

NOPREPAREFORUPGRADETOIE
Dynamically turns off the PREPAREFORUPGRADETOIE option if necessary. The default is NOPREPAREFORUPGRADETOIE.

PERFORMANCEPROFILE HIGH|MEDIUM
Valid for Extract in Integrated Capture mode.
Tunes Integrated Capture. It can be set to HIGH and MEDIUM (default). It helps achieve better performance by grouping the parameters that affect performance. Once the performance profile is set up, this option automatically configures the relevant parameters, to achieve the desired throughput and latency.

PURGEORPHANEDTRANSACTIONS | NOPURGEORPHANEDTRANSACTIONS
Valid for Extract in classic capture mode for Oracle.
Controls the purging of orphaned transactions that occur when an Oracle RAC node fails and Extract cannot capture the rollback.

PURGEORPHANEDTRANSACTIONS
Purges orphaned transactions. A transaction is verified as orphaned before purging by comparing its startup time with the node's startup time; if the transaction started earlier, it is purged.

NOPURGEORPHANEDTRANSACTIONS
The default. Orphaned transactions are not purged.

QUERYTIMEOUT *number*
Valid for SQL Server.
Specifies how long queries to SQL Server will wait for results before reporting a timeout error message. This option takes an integer value to represent the number of seconds. The default query timeout value is 30 seconds. The minimum value is 0 seconds (infinite timeout). The maximum is 2147483645 seconds.
The following example instructs SQL Server to wait 60 seconds for results before timing out.

```
TRANLOGOPTIONS QUERYTIMEOUT 60
```

QUERYRETRYCOUNT *number*
Valid for Extract for SQL Server.
Specifies how many times to retry a query to obtain table metadata after timeouts. Timeouts can occur for a long-running transaction that has created any table. The system tables become locked and prevent Extract's query from completing.
The default is one retry after a 30-second wait, after which the process abends if the retry fails. `QUERYRETRYCOUNT` can be specified to retry multiple times at 30-second intervals, according to the input value that is supplied. If all of the retries fail, Extract abends with the normal connection error message. The minimum (0) is infinite, maximum is 1000, and default is 1.
The following example causes Extract to attempt its query four times at 30-second intervals:

```
TRANLOGOPTIONS QUERYRETRYCOUNT 4
```

READQUEUESIZE *size*
Valid for MySQL.
Specifies the internal queue size, in bytes, for transaction data. It can be increased to improve performance. Valid values are integers from 1 through 214748364750000. The default is 256 bytes; start with the default and evaluate performance before adjusting upward.

REDO_TRANSPORT_LAG_THRESHOLD *seconds*
Valid for Integrated Extract in Downstream Mining Mode.
Monitors the network latency between a source database and target database when redo logs are shipped. If the latency exceeds the specified threshold then a warning appears in the report file and a subsequent information message appears when the lag drops to the normal level.
The default threshold value is 10 seconds.
For more information, see Configuring Redo Transport from Source to Downstream Mining Database

REQUIRELONGDATACAPTURECHANGES | NOREQUIRELONGDATACAPTURECHANGES
Valid for DB2 LUW.
Controls the response of Extract when `DATA CAPTURE` is set to `NONE` or to `CHANGES` without `INCLUDE LONGVAR COLUMNS` and the parameter file includes any of the following Oracle GoldenGate parameters that require the presence of before

images for some or all column values: GETBEFOREUPATES, NOCOMPRESSUPDATES, and NOCOMPRESSDELETES. Both of those DATA CAPTURE settings prevent the logging of before values for LONGVAR columns. If those columns are not available to Extract, it can affect the integrity of the target data.

REQUIRELONGDATACAPTURECHANGES
Extract abends with an error.

NOREQUIRELONGDATACAPTURECHANGES
Extract issues a warning but continues processing the data record.

SOURCE_OS_TIMEZONE *timezone*
Valid for Extract in integrated capture mode for Oracle.
Specifies the system time zone of the source database. The system time zone of a database is usually given by the default time zone of its operating system, and can also be overridden by setting the TZ environment variable when the database is started. You should specify this option only if the source database and the Extract process use different system time zones. For example, in a downstream capture deployment where the source database and the Extract process run on different servers in different time zones.
You can specify the value of this option in a time zone region name or a UTC offset form and you must use the same form used by the source database. For example, if the source database uses a region name form like America/New_York, then you must specify America/New_York, US/Eastern, or EST5EDT. Alternately, if the source database uses a UTC offset form like -05:00, then you must use the syntax (GMT) [+|-]hh[:mm]. For example, GMT-05:00 or -5.

SUPPRESSNOOOPUPDATES
Valid for Extract on Oracle Database 12*c* (12.2) and later.
You can control whether no-op updates are filtered or not in Integrated Extract. The default is no suppression.

TRANSCLEANUPFREQUENCY *minutes*
Valid for Extract in classic capture mode for Oracle.
Specifies an interval, in minutes, after which Oracle GoldenGate scans for orphaned transactions, and then scans again to delete them. The initial scan marks transactions considered to be orphaned. The second scan confirms they are orphaned, and they are deleted. Valid values are from 1 to 43200 minutes. Default is 10 minutes.

TRANCOUNT
Valid for SQL Server.
Allows you to specify a number of transactions to be fetched from the SQL Server tables at a time. It takes an integer value to represent the number of transactions. The minimum value is 1, maximum is 100, and the default is 10.
Example:

```
TRANLOGOPTIONS TRANCOUNT 20
```

This example instructs Extract to fetch 20 transactions at a time from CDC enabled tables.

[TSLOOKUPBEGINLRI | TSLOOKUPENDLRI]
Valid for DB2 LUW v10.1 and later.

When you specify an LRI range using these parameters, Extract looks for the timestamp specified in the ADD or ALTER EXTRACT command within this range. This helps Extract to optimize the look up process for a particular timestamp in the database transaction log. The TSLOOKUPBEGINLRI parameter is mandatory while TSLOOKUPENDLRI is optional. Specifying only TSLOOKUPENDLRI without TSLOOKUPBEGINLRI is invalid. For example:

```
TRANLOGOPTIONS TSLOOKUPBEGINLRI 75200.666197, TSLOOKUPENDLRI  75207.666216
TRANLOGOPTIONS TSLOOKUPBEGINLRI 75200.666197
```

If the provided timestamp falls between the given LRI ranges or the provided timestamp falls after the TSLOOKUPBEGINLRI LRI timestamp then Extract starts from a record with timestamp equal to or nearest less than the provided timestamp.
If the provided timestamp falls before TSLOOKUPBEGINLRI LRI timestamp, Extract is started from the specified TSLOOKUPBEGINLRI LRI. If the provided timestamp falls after TSLOOKUPENDLRI timestamp, then Extract abends. If you only specify TSLOOKUPENDLRI, then an informational message is displayed and Extract starts from a record with timestamp equal or nearest less than the provided timestamp.

USENATIVEOBJSUPPORT | NOUSENATIVEOBJSUPPORT
Valid for Extract in integrated capture mode for Oracle.
Integrated Capture adds redo-based capture for User Defined Type (UDT) and ANYDATA data types. It is enabled by default and can only be enabled if the source database version is 12.1.0.1 or greater and the source database compatibility is 12.0.0.0.0 or greater. Replicat from Oracle GoldenGate release 12.1.2.1.0 must be used. If a source or target database of release 12.1.0.1 is used, the Streams patch for bug 18038108 must be installed on the database. To use Native Support, all of your Oracle databases and Oracle GoldenGate instances must be release 12.1.0.1 or greater to be compatible.
If redo-based capture is enabled but a UDT contains an unsupported attribute, Integrated Capture retries to capture the UDT using fetch. For limitations of support for capture, see XML Data Types in *Using Oracle GoldenGate for Oracle Database*.
If you create object tables by using a CREATE TABLE AS SELECT (CTAS) statement, Integrated Capture must be configured to capture DML from CTAS operation in order to fully support object tables. For CTAS use information, see How Oracle GoldenGate Handles Derived Object Names in *Using Oracle GoldenGate for Oracle Database*
The default is USENATIVEOBJSUPPORT if supported.

USE_ROOT_CONTAINER_TIMEZONE
This parameter is for a CDB environment. Each PDB in a CDB can use a different database time zone. If the database time zone is available, Extract tries to get the time zone of a PDB from Integrated Dictionary. The time zone extraction requires a patch on the mining database. If the patch is not available, Extract sends a query to the PDB to get the time zone. If the database patch or a connection to the PDB is not available, and this parameter is specified, Extract assumes that the PDB database time zone is the same as the root container database time zone.

USEPREVRESETLOGSID | NOUSEPREVRESETLOGSID
Valid for Extract in classic capture mode for Oracle.
Specifies that Extract will take the previous RESETLOG id as the current branch. The default is NOUSEPREVRESETLOGSID. Supports Extract in classic capture mode when capturing in an Oracle Data Guard environment.

VALIDATEINLINESFLOB
Use this option with Classic Extract to validate inline SECUREFILE LOBs when it extracts out the LOB data from the redo column value. No value is expected after VALIDATEINLINESFLOB.

VAMCOMPATIBILITY {3}
Valid for MySQL, SQL Server, and Teradata.
Ensures the VAM module and the VAM API are using the same version of the column metadata structure. As new features are added to the VAM API, the column metadata needs enhancing with new attributes. When this occurs a new version is created, adding the new column metadata attributes to the existing ones. All database implementations other than Teradata are required to update to the latest version after a new version is added. The minimum is 1, the maximum is 1 and the default is 1.

1
A value of one means the original VAM API metadata structure is being used. This structure was originally created for Teradata, which is a separate implementation from the other databases, and only Teradata still uses this level. To maintain backwards compatibility with Extract, it may be necessary to manually set VAMCOMPATIBILITY to 1 if running an earlier version of a TAM module against later releases of Extract that contain VAM versioning. Extract abends with a message to the report file if VAMCOMPATIBILITY 1 is required and not set.

2
This VAM version added column metadata attributes for SQL Server. This value is set internally by the VAM module, so setting it manually with TRANLOGOPTIONS is not required.

3
This is the current version level. All databases other than Teradata (including SQL Server) use this level and set it internally in the VAM module, so setting it manually with TRANLOGOPTIONS is not required.

Examples

Example 1
The following specifies the location of the Oracle archived logs.

```
TRANLOGOPTIONS ALTARCHIVELOGDEST /fs1/oradata/archive/log2
```

Example 2
The following Oracle example filters for two users (one by name and one by user ID). The transactions generated by these users will be handled according to the GETREPLICATES or IGNOREREPLICATES rules, and a new transaction buffer size is specified.

```
TRANLOGOPTIONS EXCLUDEUSER ggsrep, EXCLUDEUSERID 90, BUFSIZE 100000
```

Example 3
The following shows how to deal with transaction logs that are on a platform other than the one which hosts the database.

> **Note:**
>
> The following statement spans multiple lines only because of space constraints in this documentation.

```
TRANLOGOPTIONS, LOGSOURCE VMS, PATHMAP DKA200:
[RDBMS.ORACLE.ORA9201I.64.ADMIN.GGS.ARCH]
/net/deltan/uservol1/RDBMS.DIR/ORACLE.DIR/ORA9201I.DIR/
64.DIR/admin.DIR/ggs.DIR/ARCH.dir PATHMAP DKA200:
[RDBMS.ORACLE.ORA9201I.64.ORADATA.GGS]
/net/deltan/uservol1/rdbms.dir/oracle.dir/ora9201I.DIR/
64.dir/oradata.dir/ggs.dir
```

Example 4

The following example supplies ASM credentials by specifying the alias `asm1` in the `asmdomain` domain in the Oracle GoldenGate credential store.

```
TRANLOGOPTIONS ASMUSERALIAS asm1 DOMAIN asmdomain
```

Example 5

The following is an example of how to specify the padding width when Extract forms the archive log name using the format specifiers `%T`, `%S`, and `%R` in the `ALTARCHIVELOGFORMAT` parameter.

```
TRANLOGOPTIONS ALTARCHIVELOGFORMAT ARC_%S_%R.%T
TRANLOGOPTIONS SEQPADLEN 12, RESETLOGSIDPADLEN 12, THREADPADLEN 5
```

Example 6

The following are examples of how to use tag specifiers with `EXCLUDETAG`.

```
TRANLOGOPTIONS EXCLUDETAG 00
TRANLOGOPTIONS EXCLUDETAG +
TRANLOGOPTIONS EXCLUDETAG 0952
```

Example 7

The following is an example of how to use the `TRANLOGOPTIONS FAILOVERTARGETDESTID` Extract parameter.

```
TRANLOGOPTIONS FAILOVERTARGETDESTID 2
```

```
SQL> show parameters log_archive_dest

NAME TYPE VA
LUE
---------------------------------- ----------- --
---------------------------
log_archive_dest_1 string location=USE_DB_RECOVERY_FILE_DEST,
valid_for=(ALL_LOGFILES, ALL_ROLES)
.
log_archive_dest_2 string service="ggs2d", ASYNC NOAFFIRM delay=0 optional
compression =disable max_failure=0 max_connections=1 reopen=300
db_unique_name="GGS2D" net_timeout=30,
valid_for=(online_logfile,all_roles)
```

It would be set to `2` because that is the Standby database Oracle GoldenGate should stay behind. The first entry (`log_archive_dest_1`) is for the local archive logs for that database, and the second is for the standby database.00

2.170 TRANSACTIONTIMEOUT

Valid For

Replicat

Description

Use the TRANSACTIONTIMEOUT parameter to prevent an uncommitted Replicat target transaction from holding locks on target tables and consuming database resources unnecessarily. You can change the value of this parameter so that Replicat can work within existing application timeouts and other database requirements on the target.

TRANSACTIONTIMEOUT limits the amount of time that Replicat will hold a target transaction open if it has not received the end-of-transaction record for the last source transaction in that transaction. By default, Replicat groups multiple source transactions into one target transaction to improve performance, but it will not commit a partial source transaction and will wait indefinitely for that last record. The Replicat parameter GROUPTRANSOPS controls the minimum size of a grouped target transaction. The range is 1–604800.

The following events could last long enough to trigger TRANSACTIONTIMEOUT:

- Network problems prevent trail data from being delivered to the target system.
- Running out of disk space on any system, preventing trail data from being written.
- Collector abends (a rare event).
- Extract abends or is terminated in the middle of writing records for a transaction.
- An Extract data pump abends or is terminated.
- There is a source system failure, such as a power outage or system crash.

How TRANSACTIONTIMEOUT Works

During normal operations, Replicat remembers the position in the trail of the beginning of the first source transaction in the current target transaction, in case the transaction must be abended and retried. When TRANSACTIONTIMEOUT is enabled, Replicat also saves the position of the first record of the current source transaction and will use that position as the logical end-of-file (EOF) if TRANSACTIONTIMEOUT is triggered.

When triggered, TRANSACTIONTIMEOUT does the following:

1. Aborts the current target transaction
2. Repositions to the beginning of the first source transaction in the aborted target transaction.
3. Processes all of the trail records up to the logical end-of-file position (the beginning of the last, incomplete source transaction).
4. Commits the transaction at logical EOF point.
5. Waits for new trail data before processing any more trail records.

TRANSACTIONTIMEOUT can be triggered multiple times for the same source transaction, depending on the nature of the problem that is causing the trail data to arrive slowly enough to trigger TRANSACTIONTIMEOUT.

Detecting a TRANSACTIONTIMEOUT Condition

To determine whether or not Replicat is waiting for the rest of a source transaction when TRANSACTIONTIMEOUT is enabled, issue the SEND REPLICAT command with the STATUS option. The following statuses indicate this condition:

```
Performing transaction timeout recovery
Waiting for data at logical EOF after transaction timeout recovery
```

Default

Disabled

Syntax

```
TRANSACTIONTIMEOUT n units
```

n
An integer that specifies the wait interval. Valid values are from one second to one week (seven days). This value should be greater than that set with the EOFDELAY parameter in both the primary Extract and any associated data pumps.

units
One of the following: S, SEC, SECS, SECOND, SECONDS, MIN, MINS, MINUTE, MINUTES, HOUR, HOURS, DAY, DAYS.

Example

```
TRANSACTIONTIMEOUT 5 S
```

2.171 TRANSMEMORY

Valid For

Extract for DB2 on z/OS

Description

Use the TRANSMEMORY parameter to control the amount of memory and temporary disk space available for caching uncommitted transaction data. Because Oracle GoldenGate sends only committed transactions to the target database, it requires sufficient system memory to store transaction data on the source system until either a commit or rollback indicator is received.

This parameter is for use with a DB2 database on z/OS database. For all other databases, use the CACHEMGR parameter.

About Memory Management With TRANSMEMORY

TRANSMEMORY enables you to tune the Oracle GoldenGate transaction cache size and define a temporary location on disk for storing data that exceeds the size of the cache. Options are available for defining the total cache size, the per-transaction memory size, the initial and incremental memory allocation, and disk storage space.

Transactions are added to the memory pool specified by RAM, and each is flushed to disk when TRANSRAM is reached. An initial amount of memory is allocated to each transaction based on INITTRANSRAM and is increased by the amount specified

by RAMINCREMENT as needed, up to the maximum set with TRANSRAM. Consequently, the value for TRANSRAM should be evenly divisible by the sum of (INITTRANSRAM + RAMINCREMENT).

To view current TRANSMEMORY settings, use the VIEW REPORT command in GGSCI.

Special z/OS Considerations

On a z/OS system, the RAM option not only controls the total virtual memory allocation for all cached transactions, but also controls the size of the heap memory that is allocated during startup. The large default value prevents fragmentation within the virtual memory pool, but in some installations it could cause virtual memory to be wasted, especially if the applications primarily generate small transactions. Allocating a large amount of heap memory also can cause Extract to be unresponsive at startup until z/OS completes the allocation.

On z/OS, set RAM just large enough to hold enough transaction activity without affecting the performance of Extract. If set too low, it can cause Extract to write transaction data to disk, causing Extract to run more slowly and to consume disk space. You might need to do some testing to determine the optimal value.

Default

None

Syntax

```
TRANSMEMORY
[RAM size]
[TRANSRAM size]
[TRANSALLSOURCES size]
[INITTRANSRAM size]
[RAMINCREMENT size]
[DIRECTORY (directory, max_size, max_file_size)]
```

RAM *size*
Specifies the total amount of memory to use for all cached transactions. On z/OS this also is the initial amount of memory to allocate *per transaction*. The default is 200 megabytes. The value can be specified in bytes or in terms of gigabytes, megabytes, or kilobytes in any of the following forms:
GB | MB | KB | G | M | K | gb | mb | kb | g | m | k

TRANSRAM *size*
Specifies the total amount of memory to use for a single transaction. The default is 50 megabytes. The value can be specified in bytes or in terms of gigabytes, megabytes, or kilobytes in any of the following forms:
GB | MB | KB | G | M | K | gb | mb | kb | g | m | k
TRANSRAM should be evenly divisible by both INITTRANSRAM and RAMINCREMENT for optimal results.

TRANSALLSOURCES *size*
Specifies the total amount of memory and disk space to use for a single transaction. The default is 50% of total available memory (memory and disk). The value can be specified in bytes or in terms of gigabytes, megabytes, or kilobytes in any of the following forms:
GB | MB | KB | G | M | K | gb | mb | kb | g | m | k

INITTRANSRAM *size*
(NonStop system only) Specifies the initial amount of memory to allocate for a transaction. The default is 500 kilobytes. The value can be specified in bytes or in terms of gigabytes, megabytes, or kilobytes in any of the following forms:
GB | MB | KB | G | M | K | gb | mb | kb | g | m | k

RAMINCREMENT *size*
Specifies the amount of memory to increment when a transaction requires more memory. The default is 500 kilobytes. The value can be specified in bytes or in terms of gigabytes, megabytes, or kilobytes in any of the following forms:
GB | MB | KB | G | M | K | gb | mb | kb | g | m | k

DIRECTORY (*directory, max_size, max_file_size*)
Specifies temporary disk storage for transaction data when its size exceeds the maximum specified with TRANSRAM. You can specify DIRECTORY more than once. The directory size specified with *max_size* and the file size specified with *max_file_size* must be greater than the size of the memory specified with RAM. The names of the files that are created take one of the following formats, depending on the process type:

- *group_trans*_00001.mem takes the name of the group and indicates that an online process created the file.
- *PID_trans*_00001.mem takes the name of a process ID (PID) and indicates that a one-time process (specified with the SPECIALRUN parameter) created the file.
- *group_thread#*_00001.mem takes a group name and a thread number, indicating that a threaded Extract created the file.

directory
The fully qualified path name of a directory. The default is the dirtmp sub-directory of the Oracle GoldenGate directory.

max_size
The maximum size of all files in the directory. The default is 2 gigabytes. If the space specified is not available, then 75% of available disk space is used. Values can be specified in bytes or in terms of gigabytes, megabytes, or kilobytes in any of the following forms:

GB | MB | KB | G | M | K | gb | mb | kb | g | m | k

max_file_size
The maximum size of each file in the directory. The default is 200 megabytes. Values can be specified in bytes or in terms of gigabytes, megabytes, or kilobytes in any of the following forms:

GB | MB | KB | G | M | K | gb | mb | kb | g | m | k

Examples

Example 1
The following example allows per-transaction memory to be incremented ten times before data is flushed to disk, once for the initial allocation specified with INITTRANSRAM and then nine more times as permitted by RAMINCREMENT.

```
TRANSMEMORY DIRECTORY(c:\test\dirtmp, 3000000000,
300000000), RAM 8000K, TRANSRAM 1000K, INITTRANSRAM 100K,
RAMINCREMENT 100K
```

Example 2
The following is the same as the preceding example, but with the addition of a second directory.

```
TRANSMEMORY DIRECTORY(c:\test\dirtmp, 3000000000,
300000000), DIRECTORY (c:\test\dirtmp2, 1000000000,
5000000), RAM 8000K, TRANSRAM 1000K, INITTRANSRAM 100K,
RAMINCREMENT 100K
```

> **Note:**
>
> In the previous examples, the parameter specification spans multiple lines because of space constraints. In an actual parameter file, multi-line parameter specifications must contain an ampersand (&) at the end of each line.

2.172 TRIMSPACES | NOTRIMSPACES

Valid For

Extract and Replicat

Description

Use the `TRIMSPACES` and `NOTRIMSPACES` parameters to control whether or not trailing spaces in a source `CHAR` column are truncated when applied to a target `CHAR` or `VARCHAR` column. `TRIMSPACES` and `NOTRIMSPACES` can be used at the root level of the parameter file as global `ON/OFF` switches for different sets of `TABLE` or `MAP` statements, and they can be used within an individual `TABLE` or `MAP` statement to override any global settings for that particular `MAP` or `TABLE` statement.

`TRIMSPACES` is applied only to single-byte white spaces (U+0020). Ideographic spaces (U+3000) are not supported.

For Extract, `TRIMSPACES` only has an effect if Extract is performing mapping within the `TABLE` statement (by means of a `TARGET` statement).

Default

```
TRIMSPACES
```

Syntax

```
TRIMSPACES | NOTRIMSPACES
```

Examples

Example 1
The following example uses `TRIMSPACES` and `NOTRIMSPACES` at the root level of the parameter file. The default of `TRIMSPACES` is in effect until the last `MAP` statement, to which `NOTRIMSPACES` applies.

```
MAP fin.src1, TARGET fin.tgt1;
MAP fin.src2, TARGET fin.tgt2;
MAP fin.src3, TARGET fin.tgt3;
NOTRIMSPACES
MAP fin.src4, TARGET fin.tgt4;
```

Example 2
The following example uses NOTRIMSPACES within a MAP statement to override the global default of TRIMSPACES. The default applies to the first two MAP statements, and then NOTRIMSPACES applies to the last two targets.

```
MAP fin.src1, TARGET fin.tgt1;
MAP fin.src1, TARGET fin.tgt2;
MAP fin.src1, TARGET fin.tgt3, NOTRIMSPACES;
MAP fin.src1, TARGET fin.tgt4, NOTRIMSPACES;
```

2.173 TRIMVARSPACES | NOTRIMVARSPACES

Valid For

Extract and Replicat

Description

Use the TRIMVARSPACES and NOTRIMVARSPACES parameters to control whether or not trailing spaces in a source VARCHAR column are truncated when applied to a target CHAR or VARCHAR column. TRIMVARSPACES and NOTRIMVARSPACES can be used at the root level of the parameter file as global ON/OFF switches for different sets of TABLE or MAP statements, and they can be used within an individual TABLE or MAP statement to override any global settings for that particular MAP or TABLE statement.

The default is NOTRIMVARSPACES because the spaces in a VARCHAR column can be part of the data. Before using TRIMVARSPACES, make certain that trailing spaces are not required as part of the target data.

For Extract, TRIMVARSPACES only has an effect if Extract is performing mapping within the TABLE statement (by means of a TARGET statement).

Default

```
NOTRIMVARSPACES
```

Syntax

```
TRIMVARSPACES | NOTRIMVARSPACES
```

Examples

Example 1
The following example uses TRIMVARSPACES and NOTRIMVARSPACES at the root level of the parameter file. The default of NOTRIMVARSPACES is in effect until the last MAP statement, to which TRIMVARSPACES applies.

```
MAP fin.src1, TARGET fin.tgt1;
MAP fin.src2, TARGET fin.tgt2;
MAP fin.src3, TARGET fin.tgt3;
```

```
TRIMVARSPACES
MAP fin.src4, TARGET fin.tgt4;
```

Example 2
The following example uses TRIMVARSPACES within a MAP statement to override the global default of NOTRIMVARSPACES. The default applies to the first two MAP statements, and then TRIMVARSPACES applies to the last two targets.

```
MAP fin.src1, TARGET fin.tgt1;
MAP fin.src1, TARGET fin.tgt2;
MAP fin.src1, TARGET fin.tgt3, TRIMVARSPACES;
MAP fin.src1, TARGET fin.tgt4, TRIMVARSPACES;
```

2.174 UPDATEDELETES | NOUPDATEDELETES

Valid For

Replicat

Description

Use the UPDATEDELETES parameter to convert delete operations to update operations for all MAP statements that are specified after it in the parameter file. Use NOUPDATEDELETES to turn off UPDATEDELETES. These parameters are table-specific. One remains in effect for subsequent MAP statements until the other is encountered.

Because you can selectively enable or disable these parameters between MAP statements, you can enable or disable them for different threads of a coordinated Replicat. Specify the UPDATEDELETES threads in one set of MAP statements, and specify the NOUPDATEDELETES threads in a different set of MAP statements.

When using UPDATEDELETES, use the NOCOMPRESSDELETES parameter. This parameter causes Extract to write all of the columns to the trail, so that they are available for updates.

Default

```
NOUPDATEDELETES
```

Syntax

```
UPDATEDELETES | NOUPDATEDELETES
```

Example

This example shows how you can apply UPDATEDELETES and NOUPDATEDELETES selectively to different MAP statements, each of which represents a different thread of a coordinated Replicat.

```
UPDATEDELETES
MAP sales.cust, TARGET sales.cust, THREAD (1);
MAP sales.ord, TARGET sales.ord, THREAD (2);
NOUPDATEDELETES
MAP sales.loc, TARGET sales.loc, THREAD (3);
```

2.175 UPDATEINSERTS | NOUPDATEINSERTS

Valid For

Replicat

Description

Use the UPDATEINSERTS parameter to convert insert operations to update operations for all MAP statements that are specified after it in the parameter file. Use NOUPDATEINSERTS to turn off UPDATEINSERTS.

Because you can selectively enable or disable these parameters between MAP statements, you can enable or disable them for different threads of a coordinated Replicat. Specify the UPDATEINSERTS threads in one set of MAP statements, and specify the NOUPDATEINSERTS threads in a different set of MAP statements.

Default

```
NOUPDATEINSERTS
```

Syntax

```
UPDATEINSERTS | NOUPDATEINSERTS
```

Example

This example shows how you can apply UPDATEINSERTS and NOUPDATEINSERTS selectively to different MAP statements, each of which represents a different thread of a coordinated Replicat.

```
UPDATEINSERTS
MAP sales.cust, TARGET sales.cust, THREAD (1);
MAP sales.ord, TARGET sales.ord, THREAD (2);
NOUPDATEINSERTS
MAP sales.loc, TARGET sales.loc, THREAD (3);
```

2.176 UPDATERECORDFORMAT

Valid For

Extract for all databases except Teradata.

Description

Use the UPDATERECORDFORMAT parameter to cause Extract to combine the before and after images of an UPDATE operation into a single record in the trail. It is valid for Extract in classic and integrated capture modes; it is valid for a master Extract though is not valid for pump Extract.

Before images are generated when the GETUPDATEBEFORES, GETBEFORECOLS, and LOGALLSUPCOLS parameters are used. (In the case of an update to a primary key, unique index, or user-specified KEYCOLS key, the before and after images are stored in the same record by default. UPDATERECORDFORMAT does not apply in these cases.) The NOCOMPRESSUPDATES parameter is required for non-Oracle databases.

When two records are generated for an update to a single row, it incurs additional disk I/O and processing for both Extract and Replicat. If supplemental logging is enabled on all columns, the unmodified columns may be repeated in both the before and after records. The overall size of the trail is larger, as well. This overhead is reduced by using `UPDATERECORDFORMAT`.

When `UPDATERECORDFORMAT` is used, Extract writes the before and after images to a single record that contains all of the information needed to process an `UPDATE` operation. In addition to improving the read performance of downstream processes, this enables column mapping functions to access the before and after column values at the same point in time, rather than having to cache the before image column values while reading the after values.

`UPDATERECORDFORMAT` takes effect for all `TABLE` statements in the parameter file.

If you specify both `UPDATERECORDFORMAT` and `FORMAT RELEASE 11.x` or earlier, then Extract will abend.

> **Note:**
>
> Many-columned tables can cause the trail record to reach its maximum size when `UPDATERECORDFORMAT` is used. The rest of the record is continued in one or more additional, chained record fragments. This has a minor effect on processing performance.

Default

```
UPDATERECORDFORMAT COMPACT
```

Syntax

```
UPDATERECORDFORMAT [FULL | COMPACT]
```

`FULL`
Generates one trail record that contains the before and after images of an `UPDATE`, where the before image includes all of the columns that are available in the transaction record for both the before and after images. When viewed in the Logdump utility, this record appears as `GGSUnifiedUpdate`.

`COMPACT`
Generates one trail record that contains the before and after images of an `UPDATE`, where the before image includes all of the columns that are available in the transaction record, but the after image is limited to the primary key columns and the columns that were modified in the `UPDATE`. `UPDATERECORDFORMAT COMPACT` is recommended for configurations that include an integrated Replicat. This is the default.
When either `FULL` or `COMPACT` are viewed in the Logdump utility, the record appears as `GGSUnifiedUpdate`. The record contains the following:

- a header
- the length of the before image
- the before values of each column

- the after values of the primary key, unique index, or KEYCOLS columns
- the after values of the modified columns
- internal token data

Example

```
UPDATERECORDFORMAT COMPACT
```

2.177 UPREPORT

Valid For

Manager

Description

Use the UPREPORTMINUTES or UPREPORTHOURS parameter to specify the frequency with which Manager reports Extract and Replicat processes that are running. Every time one of those processes starts or stops, events are generated. Those messages are easily overlooked in the error log because the log can be so large. UPREPORTMINUTES and UPREPORTHOURS report on a periodic basis to ensure that you are aware of the process status.

If UPREPORT is explicitly indicated and the value of the CHECKMINUTES parameter is greater than that of UPREPORT, then CHECKMINUTES acquires the value of UPREPORT.

To report on stopped processes, use the DOWNREPORT parameter. See "DOWNREPORT" for more information.

Default

Do not report running processes

Syntax

```
UPREPORTMINUTES minutes | UPREPORTHOURS hours
```

UPREPORTMINUTES *minutes*
Sets the report frequency in minutes. The minimum is 0.

UPREPORTHOURS *hours*
Sets the report frequency in hours. The minimum is 0.

Example

The following generates a report every 30 minutes.

```
UPREPORTMINUTES 30
```

2.178 USE_TRAILDEFS | NO_USE_TRAILDEFS

Valid For

Extract data pump and Replicat when used in a GLOBALS file

Description

Use the USE_TRAILDEFS and NO_USE_TRAILDEFS parameters to control where the data pump and Replicat processes obtain the table definitions when the trail files contain full table definitions.

USE_TRAILDEFS forces these processes to use the table definitions from the trail unless the OVERRIDE keyword is specified with SOURCEDEFS or ASSUMETARGETDEFS.

NO_USE_TRAILDEFS forces these processes to follow the old behavior when resolving the table definitions. Extract and pump will not generate trail files with full table definition.

Default

USE_TRAILDEFS

Syntax

```
[ USE_TRAILDEFS | NO_USE_TRAILDEFS ]
```

2.179 USEDEDICATEDCOORDINATIONTHREAD

Valid For

Replicat (coordinated mode)

Description

Use USEDEDICATEDCOORDINATIONTHREAD to force Replicat to maintain a dedicated coordination thread to apply barrier transactions. The thread ID of this thread is always 0.

By default, Replicat uses the thread with the lowest thread ID to apply barrier transactions, but that thread also includes work that is mapped to it explicitly. By using a dedicated thread for barrier transactions, you can get an accurate view in Oracle GoldenGate statistics of the number of barrier events and exposes the amount of work that is performed serially. Coordinated Replicat statistics are written to the report file and also can be viewed with the STATS REPLICAT command.

USEDEDICATEDCOORDINATIONTHREAD applies to the Replicat group as a whole, across all MAP statements.

Syntax

```
USEDEDICATEDCOORDINATIONTHREAD
```

Example

```
USEDEDICATEDCOORDINATIONTHREAD
MAP u1.t1, TARGET u2.t1 SQLEXEC &
(ID test2, QUERY 'delete from u2.t2 where col_val =100 ',  &
NOPARAMS)), THREAD(1), COORDINATED;
```

2.180 USEIPV4

Valid For

GLOBALS

Description

Use the `USEIPV4` parameter to force the use of Internet Protocol version 4 (IPv4) by Oracle GoldenGate for TCP/IP connections. By default, Oracle GoldenGate uses IPv6 in dual-stack mode and this parameter forces the use of IPv4 only.

When `USEIPV4` is used, the entire network in which Oracle GoldenGate operates must be IPv4 compatible.

Default

Disabled

Syntax

```
USEIPV4
```

2.181 USEIPV6

Valid For

GLOBALS

Description

Use the `USEIPV6` parameter to force the use of Internet Protocol version 6 (IPv6) by Oracle GoldenGate for TCP/IP connections. By default, Oracle GoldenGate uses IPv6 in dual-stack mode but falls back to IPv4, and only then to IPv6. `USEIPV6` eliminates the IPv4 fallback step. The order of socket selection becomes:

- IPv6 dual-stack
- IPv6

When `USEIPV6` is used, the entire network in which Oracle GoldenGate operates must be IPv6 compatible.

Default

Disabled

Syntax

```
USEIPV6
```

2.182 USERID | NOUSERID

Valid For

Manager, Extract, Replicat, DEFGEN

Supported for

DB2 for i

DB2 LUW

DB2 on z/OS

Oracle

MySQL

SQL Server

Teradata

Description

Use the `USERID` parameter to specify the type of authentication for an Oracle GoldenGate process to use when logging into a database, and to specify password encryption information. This parameter can be used instead of `USERIDALIAS` when an Oracle GoldenGate credential store is not being used.

Always use `USERID` or `USERIDALIAS` for a primary Extract and for Replicat. Use `USERID` or `USERIDALIAS` for Manager only if using parameters that require Manager to log into the source or target database.

USERID Compared to USERIDALIAS

`USERID` requires either specifying the clear-text password in the parameter file or encrypting it with the `ENCRYPT PASSWORD` command and, optionally, storing an encryption key in an `ENCKEYS` file. `USERID` supports a broad range of the databases that Oracle GoldenGate supports.

`USERIDALIAS` enables you to specify an alias, rather than a user ID and password, in the parameter file. The user IDs and encrypted passwords are stored in a credential store. `USERIDALIAS` supports databases running on Linux, UNIX, and Windows platforms.

General Requirements for USERID

Specify `USERID` before any `TABLE` or `MAP` entries in an Extract or Replicat parameter file. Specify `USERID` in a Manager parameter file if Manager must access the database and a login is required.

`USERID` is not always required, nor is `PASSWORD` always required when `USERID` is required. In some cases, it is sufficient just to use `USERID` or even just to use the `SOURCEDB` or `TARGETDB` parameter, depending on how authentication for the database is configured.

See "SOURCEDB" and "TARGETDB" for more information.

> **Note:**
>
> The privileges that are required for the USERID user vary by database. See the appropriate Oracle GoldenGate installation guide for your database to determine the privileges that are required for the Oracle GoldenGate database users.

USERID Requirements Per Database Type

The usage of USERID varies depending on the database type.

DB2 for i

Use USERID with PASSWORD to specify the name and password of the user profile assigned to the Oracle GoldenGate process. Use SOURCEDB or TARGETDB with USERID to specify the default DB2 for i database that is identified by the system name (in upper case). See *Using Oracle GoldenGate for Heterogeneous Databases* for more information.

DB2 for LUW

Use USERID with PASSWORD and preceded by SOURCEDB or TARGETDB for all Oracle GoldenGate processes that connect to a DB2 LUW database using database authentication. You can omit USERID and PASSWORD (and only use SOURCEDB or TARGETDB) if the database is configured allow authentication at the operating-system level.

DB2 for z/OS database

Use USERID with PASSWORD if the user that is assigned to the Oracle GoldenGate process does not have the DB2 privileges that are required for the process to function properly.

MySQL

Use USERID with PASSWORD for all Oracle GoldenGate processes that connect to a MySQL database.

Oracle

Use USERID for Oracle GoldenGate processes that connect to an Oracle database as follows:

- To use an operating system login, use USERID with the / argument.
- To use a database user name and password, use USERID with PASSWORD.
- Optionally, you can specify the user to log in as sysdba.
- (*Oracle Enterprise Edition earlier than 11.2.0.2*) Special database privileges are required for the USERID user when Extract is configured to use LOGRETENTION. These privileges might have been granted when Oracle GoldenGate was installed. See Log Retention Options in *Using Oracle GoldenGate for Oracle Database*for LOGRETENTION information.

- (*Oracle Standard or Enterprise Edition 11.2.0.2 or later*) To use `USERID` for an Extract group that is configured for integrated capture, the user must have the privileges granted in the `dbms_goldengate_auth.grant_admin_privilege` procedure, and the user must be the same one that issues `DBLOGIN` and `REGISTER EXTRACT` or `UNREGISTER EXTRACT` for the Extract group that is associated with this `USERID`.
- To support capture from an Oracle container database, the user that is specified with `USERID` must log into the root container and must be a common user. A connect string must be supplied for this user and must include the required `C##` prefix of the common user, such as `C##GGADMIN@FINANCE`. See, Other Requirements for Multitenant Container Databases in *Using Oracle GoldenGate for Oracle Database*.
- If you are connecting a Replicat into a multitenant database that each Replicat connects into a single PDB and if your are applying data into multiple PDBs in the same multitenant database, then Oracle recommends that you use multiple Replicats.

Use `NOUSERID` to allow Integrated Extract to run without a connection for fetching or metadata lookups, or any data dictionary calls. Essentially eliminating the need to connect to the source database at all. The `NOUSERID` option requires an Integrated Dictionary. We should also include that when `NOUSERID` is used, if the customer has an Active Data Guard Standby, they can set up fetching from that Standby database using the `FETCHUSERID` parameter. The two can be used in conjunction with `NOUSERID`. In the event where you are using downstream integrated extract (same caveats below) you can use `FETCHUSERID` to fetch from the ADG Standby database and `NOUSERID` to prevent the Extract from making a connection to the source database. This way, if Extract does need to fetch, it can do so.

SQL Server

Use `USERID` with `PASSWORD` if the ODBC data source connection that will be used by the Oracle GoldenGate process is configured to connect using SQL Server.

- On a source SQL Server system, also use the `SOURCEDB` parameter to specify the source ODBC data source.
- On a target SQL Server system, also use the `TARGETDB` parameter to specify the target ODBC data source.

Teradata

Use `USERID` with `PASSWORD` for Oracle GoldenGate processes that connect to a Teradata database.

- On a source Teradata system, also use the `SOURCEDB` parameter to specify the source ODBC data source.
- On a target Teradata system, also use the `TARGETDB` parameter to specify the target ODBC data source.

Default

None

Syntax

```
USERID {/ | user}[, PASSWORD password]
[algorithm ENCRYPTKEY {key_name | DEFAULT}] [SYSDBA]
[, THREADS (threadID[, threadID][, ...][, thread_range[, thread_range]
[, ...])]
```

/
Directs Oracle GoldenGate to use an operating-system login for Oracle, not a database user login. Use this argument only if the database allows authentication at the operating-system level. Bypassing database-level authentication eliminates the need to update Oracle GoldenGate parameter files if application passwords frequently change. To use this option, the correct user name must exist in the database, in relation to the value of the Oracle `OS_AUTHENT_PREFIX` initialization parameter, as follows:

- The value set with `OS_AUTHENT_PREFIX` is concatenated to the beginning of a user's operating system account name and then compared to the database name. Those two names must match.
- If `OS_AUTHENT_PREFIX` is set to `' '` (a null string), the user name must be created with `IDENTIFIED EXTERNALLY`. For example, if the OS user name is `ogg`, you would use the following to create the database user:

  ```
  CREATE USER ogg IDENTIFIED EXTERNALLY;
  ```

- If `OS_AUTHENT_PREFIX` is set to `OPS$` or another string, the user name must be created in the following format:

  ```
  OS_AUTHENT_PREFIX_value OS_user_name
  ```

  For example, if the OS user name is `ogg`, you would use the following to create the database user:

  ```
  CREATE USER ops$ogg IDENTIFIED BY oggpassword;
  ```

user
Specifies the name of a database user or a schema, depending on the database configuration. For Oracle, a SQL*Net connect string can be used. Refer to USERID Requirements Per Database Type for additional guidelines.

password
Use when database authentication is required to specify the password for the database user. If the password was encrypted by means of the `ENCRYPT PASSWORD` command, supply the encrypted password; otherwise, use the clear-text password. If the password is case-sensitive, type it that way.
If either the user ID or password changes, the change must be made in the Oracle GoldenGate parameter files, including the re-encryption of the password if necessary.

algorithm
Specifies the encryption algorithm that was used to encrypt the password with `ENCRYPT PASSWORD`.
The algorithm can be one of:
`AES128`
`AES192`
`AES256`
`BLOWFISH`

ENCRYPTKEY {*key_name* | DEFAULT}
Specifies the encryption key that was specified with ENCRYPT PASSWORD.

- ENCRYPTKEY *key_name* specifies the logical name of a user-created encryption key in the ENCKEYS lookup file. Use if ENCRYPT PASSWORD was used with the *KEYNAME key_name* option.
- ENCRYPTKEY DEFAULT directs Oracle GoldenGate to use a random key. Use if ENCRYPT PASSWORD was used with the KEYNAME DEFAULT option.

SYSDBA
(Oracle) Specifies that the user logs in as sysdba.

THREADS (*threadID*[, *threadID*][, ...][, *thread_range*[, *thread_range*][, ...])
Valid for Replicat. Links the specified credential to one or more threads of a coordinated Replicat. Enables you to specify different logins for different threads.

***threadID*[, *threadID*][, ...]**
Specifies a thread ID or a comma-delimited list of threads in the format of threadID, threadID, threadID.

[, *thread_range*[, *thread_range*][, ...]
Specifies a range of threads in the form of threadIDlow-threadIDhigh or a comma-delimited list of ranges in the format of threadIDlow-threadIDhigh, threadIDlow-threadIDhigh.

A combination of these formats is permitted, such as threadID, threadID, threadIDlow-threadIDhigh.

Examples

Example 1

```
USERID /
```

Example 2

```
USERID ogg
```

Example 3

```
USERID ogg@ora1.ora, &
PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC AES128, &
ENCRYPTKEY securekey1
```

Example 4

```
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
AES128, ENCRYPTKEY securekey1
```

Example 5

```
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
BLOWFISH, ENCRYPTKEY DEFAULT
```

Example 6

```
USERID ogg, &
PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC AES128, &
ENCRYPTKEY securekey1 SYSDBA
```

2.183 USERIDALIAS

Valid For

Manager, Extract, Replicat, DEFGEN

Supported for

DB2 for i

DB2 LUW

DB2 on z/OS

MySQL

Oracle

SQL Server

Teradata

Description

Use the `USERIDALIAS` parameter to specify authentication for an Oracle GoldenGate process to use when logging into a database. The use of `USERIDALIAS` requires the use of an Oracle GoldenGate credential store. Specify `USERIDALIAS` before any `TABLE` or `MAP` entries in the parameter file.

> **Note:**
>
> The privileges that are required for the `USERIDALIAS` user vary by database. See the *Using Oracle GoldenGate for Heterogeneous Databases*to determine the privileges that are required for the Oracle GoldenGate database users.

USERIDALIAS Compared to USERID

`USERIDALIAS` enables you to specify an alias, rather than a user ID and password, in the parameter file. The user IDs and encrypted passwords are stored in a credential store. `USERIDALIAS` supports databases running on Linux, UNIX, and Windows platforms.

`USERID` requires either specifying the clear-text password in the parameter file or encrypting it with the `ENCRYPT PASSWORD` command and, optionally, storing an encryption key in an `ENCKEYS` file. `USERID` supports a broad range of the databases that Oracle GoldenGate supports. In addition, it supports the use of an operating system login for Oracle databases.

USERIDALIAS Requirements Per Database Type

The usage of `USERIDALIAS` varies depending on the database type.

> **Note:**
>
> Logins that require a database user and password must be stored in the Oracle GoldenGate credential store.

DB2 for LUW

Use `USERIDALIAS` with the `SOURCEDB` or `TARGETDB` parameter for all Oracle GoldenGate processes that connect to a DB2 LUW database using database authentication. You can omit `USERIDALIAS` and only use `SOURCEDB` or `TARGETDB` if the database is configured allow authentication at the operating-system level. See "SOURCEDB" and "TARGETDB" for more information.

MySQL

Use `USERIDALIAS` for all Oracle GoldenGate processes that connect to a MySQL database. The `SOURCEDB` or `TARGETDB` parameter is not required.

Oracle

Use `USERIDALIAS` for Oracle GoldenGate processes that connect to an Oracle database.

- The `SOURCEDB` or `TARGETDB` parameter is not required.
- Specify the alias of a database credential that is stored in the Oracle GoldenGate credential store.
- (*Oracle Enterprise Edition earlier than 11.2.0.2*) Special database privileges are required for the `USERIDALIAS` user when Extract is configured to use `LOGRETENTION`. These privileges might have been granted when Oracle GoldenGate was installed. See the Log Retention Options in *Using Oracle GoldenGate for Oracle Database* for more information about `LOGRETENTION`.
- (*Oracle Standard or Enterprise Edition 11.2.0.2 or later*) To use `USERIDALIAS` for an Extract group that is configured for integrated capture, the user must have the privileges granted in the `dbms_goldengate_auth.grant_admin_privilege` procedure, and the user must be the same one that issues `DBLOGIN` and `REGISTER EXTRACT` or `UNREGISTER EXTRACT` for the Extract group that is associated with this `USERIDALIAS`.
- To support capture from an Oracle container database, the user that is specified with `USERID` must log on to the root container and must be a common database user. A connect string must be supplied for this user, for example: `C##GGADM@FINANCE`. For more information, see Configuring the Primary Extract in *Using Oracle GoldenGate for Heterogeneous Databases*.

SQL Server

Use `USERIDALIAS` if the ODBC data source connection that will be used by the Oracle GoldenGate process is configured to supply database authentication. `USERIDALIAS` can be a specific login that is assigned to the process or any member of an account in the `System Administrators` or `Server Administrators` fixed server role.

- On a source SQL Server system, also use the `SOURCEDB` parameter to specify the source ODBC data source.

- On a target SQL Server system, also use the TARGETDB parameter to specify the target ODBC data source.

Teradata

Use USERIDALIAS for Oracle GoldenGate processes that connect to a Teradata database.

- On a source Teradata system, also use the SOURCEDB parameter to specify the source ODBC data source.
- On a target Teradata system, also use the TARGETDB parameter to specify the target ODBC data source.

Default

None

Syntax

```
USERIDALIAS alias [DOMAIN domain] [SYSDBA]
[, THREADS (threadID[, threadID][, ...][, thread_range[, thread_range][, ...])]
```

alias
Specifies the alias of a database user credential that is stored in the Oracle GoldenGate credential store. Refer to USERID Requirements Per Database Type for additional guidelines.

DOMAIN *domain*
Specifies the credential store domain for the specified alias. A valid domain entry must exist in the credential store for the specified alias.

SYSDBA
(Oracle) Specifies that the user logs in as sysdba.

THREADS (*threadID*[, *threadID*][, ...][, *thread_range*[, *thread_range*][, ...])
Valid for Replicat. Links the specified credential to one or more threads of a coordinated Replicat. Enables you to specify different logins for different threads.

***threadID*[, *threadID*][, ...]**
Specifies a thread ID or a comma-delimited list of threads in the format of threadID, threadID, threadID.

[, *thread_range*[, *thread_range*][, ...]
Specifies a range of threads in the form of threadIDlow-threadIDhigh or a comma-delimited list of ranges in the format of threadIDlow-threadIDhigh, threadIDlow-threadIDhigh.

A combination of these formats is permitted, such as threadID, threadID, threadIDlow-threadIDhigh.

Examples

Example 1
The following supplies a credential for the user in the credential store that has the alias of tiger1 in the domain of east.

Example 2
The following supplies a credential for thread 3 of a coordinated Replicat.

```
USERIDALIAS tiger1 DOMAIN east THREADS (3)
```

2.184 VAM

Valid For

Extract

Description

Use the `VAM` parameter to specify that a Vendor Access Module (VAM) is being used to perform data capture functions for the Extract process and send it to the Extract API. This parameter supplies required input for the VAM API.

Default

None

Syntax

```
VAM library, PARAMS ('param' [, 'param'] [, ...])
```

library
The name of the library that is supplied by the database vendor as a Windows DLL or a UNIX shared object. Use the full path name if the library is in a directory other than the Oracle GoldenGate directory.

> **Note:**
>
> Teradata calls this library the *Teradata Access Module (TAM)*. This program or library communicates with the Oracle GoldenGate VAM API.

PARAMS *'param'*
Any parameter, enclosed within single quotes, that is passed to the Oracle GoldenGate API. See the following database-specific parameter options.

inifile, *ini_file*, callbackLib, extract.exe
Required parameter for Teradata.

- `inifile` indicates that the next parameter specifies the TAM initialization file.
- *`ini_file`* is the name of the TAM initialization file. Unless the file resides in the same directory where the Extract program is installed, specify the fully qualified path name.
- `callbackLib` indicates that the next parameter specifies the program that interfaces with the TAM. This parameter is case-sensitive and must be entered exactly as shown here.

- `extract.exe` is the Extract program, which is the callback program for the TAM.

Examples

```
VAM tam.dll, PARAMS (inifile, tam.ini, callbackLib, extract.exe)
VAM PARAMS(CDCRECORDSFETCHCOUNT 5)
VAM PARAMS(CDCRECORDSFETCHCOUNT 5 CDCSESSIONTIMEOUT 5)
VAM PARAMS (CDCRECORDQUEUESIZE 512)
VAM PARAMS (LOGICALLOGWARNINGTHRESHOLD 65)
```

2.185 VARWIDTHNCHAR | NOVARWIDTHNCHAR

Valid For

Extract, Replicat, DEFGEN for Oracle

Description

Use the `VARWIDTHNCHAR` and `NOVARWIDTHNCHAR` parameters to control how `NCHAR` data is written to the trail and interpreted by Replicat.

- `VARWIDTHNCHAR` causes an `NCHAR`, `NVARCHAR2`, or `NCLOB` character set to be treated as a variable-length character set (UTF-8).
- `NOVARWIDTHNCHAR` causes an `NCHAR`, `NVARCHAR2`, or `NCLOB` character set to be treated as UTF-16.
- If neither option is specified, the `NLS_NCHAR_CHARACTERSET` property value from the database is used to determine how an `NCHAR`, `NVARCHAR2`, or `NCLOB` character set is treated.

Default

Use `NLS_NCHAR_CHARACTERSET` property from database

Syntax

```
VARWIDTHNCHAR | NOVARWIDTHNCHAR
```

2.186 WALLETLOCATION

Valid For

GLOBALS

Description

Use the `WALLETLOCATION` parameter to specify the location of the Oracle GoldenGate master-key wallet.

Default

The `dirwlt` subdirectory of the Oracle GoldenGate installation directory.

Syntax

```
WALLETLOCATION directory_path
```

directory_path
Specifies the full path name of the wallet location.

Example

```
WALLETLOCATION /home/ggadmin/walletdir
```

2.187 WARNLONGTRANS

Valid For

Extract

Description

Use the WARNLONGTRANS parameter to specify a length of time that a transaction can be open before Extract generates a warning message that the transaction is long-running. Also use WARNLONGTRANS to control the frequency with which Oracle GoldenGate checks for long-running transactions.

This parameter is valid for Oracle and SQL Server.

When WARNLONGTRANS is specified, Oracle GoldenGate checks for transactions that satisfy the specified threshold, and it reports the first one that it finds to the Oracle GoldenGate error log, the Extract report file, and the system log. By default, Oracle GoldenGate repeats this check every five minutes.

To view a list of open transactions on demand, to output transaction details to a file, or to either cancel those transactions or force them to the trail, use the options of the SEND EXTRACT command.

Default

One hour (and check every five minutes using a separate processing thread)

Syntax

```
WARNLONGTRANS duration
[, CHECKINTERVAL interval]
[, NOUSETHREADS]
[, USELASTREADTIME]
```

duration
Sets a length of time after which an open transaction is considered to be long-running. The duration is specified as a whole number, followed by the unit of time in any of the following formats to indicate seconds, minutes, or hours. Do not put a space between the numeric value and the unit of time. The unit is not case-sensitive. The default is one hour.

```
S|SEC|SECS|SECOND|SECONDS
M|MIN|MINS|MINUTE|MINUTES
H|HOUR|HOURS
D|DAY|DAYS
```

The following are examples of valid durations:

```
WARNLONGTRANS 1DAY
WARNLONGTRANS 600sec
WARNLONGTRANS 40s
```

CHECKINTERVAL *interval*
Sets the frequency at which Oracle GoldenGate checks for transactions that satisfy WARNLONGTRANS and reports the longest running one. The interval is specified as a whole number, followed by the unit of time in any of the following formats to indicate seconds, minutes, or hours. Do not put a space between the numeric value and the unit of time. The unit is not case-sensitive. The default is five minutes, which is also the minimum valid value. The minimum value is 300 and the maximum is 20000000.

```
S|SEC|SECS|SECOND|SECONDS
M|MIN|MINS|MINUTE|MINUTES
H|HOUR|HOURS
D|DAY|DAYS

CHECKINTERVAL 1day
CHECKINTERVAL 600SEC
CHECKINTERVAL 2m
```

NOUSETHREADS
Valid for Oracle.
Specifies that the monitoring will be done by the main process thread. By default, it is done with a separate thread for performance reasons. NOUSETHREADS should only be used if the system does not support multi-threading.

USELASTREADTIME
Valid for Oracle.
Forces Extract to always use the time that it last read the Oracle redo log to determine whether a transaction is long-running or not. By default, Extract uses the timestamp of the last record that it read from the redo log. This applies to an Extract that is running in archive log only mode, as configured with TRANLOGOPTIONS using the ARCHIVEDLOGONLY option.

Example

```
NOUSETHREADS
```

2.188 WARNRATE

Valid For

Replicat

Description

Use the WARNRATE parameter to set a threshold for the number of SQL errors that can be tolerated on any target table before being reported to the process report and to the error log. The errors are reported as a warning. If your environment can tolerate a large number of these errors, increasing WARNRATE helps to minimize the size of those files.

When setting WARNRATE for a coordinated Replicat, take into account that the specified WARNRATE threshold is applied to each thread in the configuration, not as an aggregate

threshold for Replicat as a whole. For example, if `WARNRATE 100` is specified, it is possible for each thread to return 99 errors without a warning from Replicat.

For Replicat running in an Oracle environment, this parameter is valid for nonintegrated mode only.

Default

100 errors

Syntax

```
WARNRATE number_of_errors
```

number_of_errors
The number of SQL errors after which a warning is issued.

Example

```
WARNRATE 1000
```

2.189 WILDCARDRESOLVE

Valid For

Extract and Replicat

Description

Use the `WILDCARDRESOLVE` parameter to alter the rules for processing wildcarded table specifications in a `TABLE`, `SEQUENCE`, or `MAP` statement. `WILDCARDRESOLVE` must precede the associated `TABLE`, `SEQUENCE`, or `MAP` statements in the parameter file.

The target objects must already exist in the target database when wildcard resolution is attempted. If a target object does not exist, Replicat abends.

Default

```
DYNAMIC
```

Syntax

```
WILDCARDRESOLVE {DYNAMIC | IMMEDIATE}
```

`DYNAMIC`
Source objects that satisfy the wildcard definition are resolved each time the wildcard rule is satisfied. The newly resolved object is included in the Oracle GoldenGate configuration upon resolution. This is the default. This is the required setting for Teradata.
Do not use this option when `SOURCEISTABLE` or `GENLOADFILES` is specified. `WILDCARDRESOLVE` will always be implicitly set to `IMMEDIATE` for these parameters. `DYNAMIC` must be used when using wildcards to replicate Oracle sequences with the `SEQUENCE` parameter.
To keep the default of `DYNAMIC`, an explicit `WILDCARDRESOLVE` parameter is optional, but its presence helps make it clear to someone who is reviewing the parameter file which method is being used.

IMMEDIATE
Source objects that satisfy the wildcard definition are processed at startup. This option is not supported for Teradata. This is the forced default for SOURCEISTABLE.
This option does not support the Oracle interval partitioning feature. Dynamic resolution is required so that new partitions are found by Oracle GoldenGate.

Example

The following example resolves wildcards at startup.

```
WILDCARDRESOLVE IMMEDIATE
TABLE hq.acct_*;
```

2.190 XAGENABLE

Valid For

GLOBALS

Description

Use XAGENABLE to enable the Oracle GoldenGate Transparent Integration with Clusterware feature that allows you to continue using GGSCI to start and stop manager when GoldenGate instance is under the management of Oracle Grid Infrastructure Bundled Agents (XAG). You must set one of the following environment variables when using XAGENABLE:

```
CRS_HOME
ORA_CRS_HOME
GRID_HOME
```

You can use INFO ALL to view XAG related information.

Default

Disabled.

Syntax

```
XAGENABLE
```

2.191 Y2KCENTURYADJUSTMENT | NOY2KCENTURYADJUSTMENT

Valid For

Extract and Replicat

Description

Use the Y2KCENTURYADJUSTMENT and NOY2KCENTURYADJUSTMENT parameters to control the conversion of year values when the century portion consists of zeroes (such as 0055) or is not specified (such as in a two-digit, year-only specification).

With Y2KCENTURYADJUSTMENT enabled (the default), a two-digit year value that is greater than or equal to 50 is converted to a four-digit year in the 20th century (19xx). If a two-digit year value is less than 50, it is converted to a four-digit year in the 21st century (20xx).If the century portion of the year is non-zero, or if NOY2KCENTURYADJUSTMENT is specified, no conversion is performed.

Default

Y2KCENTURYADJUSTMENT

Syntax

```
Y2KCENTURYADJUSTMENT | NOY2KCENTURYADJUSTMENT
```

3
Column Conversion Functions

The column conversion functions of Oracle GoldenGate enable you to manipulate source values into the appropriate format for target columns.You can manipulate numbers and characters, perform tests, extract parameter values, return environment information, and more. For more information about using these functions, see *Administering Oracle GoldenGate*.

Topics:

- Summary of Column-Conversion Functions
  This summary is organized according to the types of processing that can be performed with the Oracle GoldenGate functions.
- @RANGE
- AFTER
- BEFORE
- BEFOREAFTER
- BINARY
- BINTOHEX
- CASE
- COLSTAT
- COLTEST
- COMPUTE
- DATE
- DATEDIFF
- DATENOW
- DDL
- EVAL
- GETENV
- GETVAL
- HEXTOBIN
- HIGHVAL | LOWVAL
- IF
- NUMBIN
- NUMSTR
- OGG_SHA1
- STRCAT
- STRCMP

- STRCMPNULL
- STREQ
- STREQNULL
- STREXT
- STRFIND
- STRLEN
- STRLTRIM
- STRNCAT
- STRNCMP
- STRNUM
- STRRTRIM
- STRSUB
- STRTRIM
- STRUP
- TOKEN
- VALONEOF

3.1 Summary of Column-Conversion Functions

This summary is organized according to the types of processing that can be performed with the Oracle GoldenGate functions.

These functions are used to perform tests.

Function	Description
CASE	Selects a value depending on a series of value tests.
EVAL	Selects a value based on a series of independent tests.
IF	Selects one of two values depending on whether a conditional statement returns TRUE or FALSE.

These functions handle missing columns.

Function	Description
COLSTAT	Returns an indicator that a column is `MISSING`, `NULL`, or `INVALID`.
COLTEST	Performs conditional calculations to test whether a column is `PRESENT`, `MISSING`, `NULL`, or `INVALID`.

These functions work with dates.

Function	Description
DATE	Returns a date and time based on the format passed into the source column.

Function	Description
DATEDIFF	Returns the difference between two dates or datetimes.
DATENOW	Returns the current date and time.

These functions are used to perform arithmetic calculations.

Function	Description
COMPUTE	Returns the result of an arithmetic expression.

These functions work with strings.

Function	Description
NUMBIN	Converts a binary string into a number.
NUMSTR	Converts a string into a number.
STRCAT	Concatenates one or more strings.
STRCMP	Compares two strings.
STREXT	Extracts a portion of a string.
STREQ	Determines whether or not two strings are equal.
STRFIND	Finds the occurrence of a string within a string.
STRLEN	Returns the length of a string.
STRLTRIM	Trims leading spaces.
STRNCAT	Concatenates one or more strings to a maximum length.
STRNCMP	Compares two strings based on a specified number of characters.
STRNUM	Converts a number into a string.
STRRTRIM	Trims trailing spaces.
STRSUB	Substitutes one string for another.
STRTRIM	Trims leading and trailing spaces.
STRUP	Changes a string to uppercase.
VALONEOF	Compares a string or string column to a list of values.

These are miscellaneous functions.

Function	Description
AFTER	Returns the after image of the specified column.
BEFORE	Returns the before image of the specified column.
BEFOREAFTER	Returns the before image of the specified column, if available, otherwise returns the after image.
BINARY	Maintains source binary data as binary data in the target column when the source column is defined as a character column.
BINTOHEX	Converts a binary string to a hexadecimal string.
GETENV	Returns environmental information.

Function	Description
GETVAL	Extracts parameters from a stored procedure as input to a FILTER or COLMAP clause.
HEXTOBIN	Converts a hexadecimal string to a binary string.
HIGHVAL | LOWVAL	Constrains a value to a high or low value.
@RANGE	Divides rows into multiple groups of data for parallel processing.
TOKEN	Retrieves token data from a trail record header.
OGG_SHA1	Hashes some fields while replicating them to Operational Data Store.

3.2 @RANGE

Use the @RANGE function to divide the rows of any table across two or more Oracle GoldenGate processes. It can be used to increase the throughput of large and heavily accessed tables and also can be used to divide data into sets for distribution to different destinations. Specify each range in a FILTER clause in a TABLE or MAP statement.

@RANGE is safe and scalable. It preserves data integrity by guaranteeing that the same row will always be processed by the same process group. To ensure that rows do not shift partitions to another process group and that the DML is performed in the correct order, the column on which you base the @RANGE partitioning must not ever change during a process run. Updates to the partition column may result in "row not found" errors or unique-constraint errors.

@RANGE computes a hash value of the columns specified in the input. If no columns are specified, the KEYCOLS clause of the TABLE or MAP statement is used to determine the columns to hash, if a KEYCOLS clause exists. Otherwise, the primary key columns are used.

Oracle GoldenGate adjusts the total number of ranges to optimize the even distribution across the number of ranges specified.

Because any columns can be specified for this function, rows in tables with relational constraints to one another must be grouped together into the same process or trail to preserve referential integrity.

> **Note:**
>
> Using Extract to calculate the ranges is more efficient than using Replicat. Calculating ranges on the target side requires Replicat to read through the entire trail to find the data that meets each range specification.

Syntax

```
@RANGE (range, total_ranges [, column] [, column] [, ...])
```

range
The range assigned to the specified process or trail. Valid values are 1, 2, 3, and so forth, with the maximum value being the value defined by *total_ranges*.

total_ranges
The total number of ranges allocated. For example, to divide data into three groups, use the value 3.

column
The name of a column on which to base the range allocation. This argument is optional. If not used, Oracle GoldenGate allocates ranges based on the table's primary key.
Your data cannot contain missing or NULL columns; this will cause the @RANGE function to abend.
The columns specified in a list of columns must exist in the trail file. You can control this using KEYCOLS in the Extract to include this column, or by using FETCHCOLS in the Extract for the column, or by ensuring that the column is part of the supplemental log group and then using LOGALLSUPCOLS.

Examples

Example 1
In the following example, the replication workload is split into three ranges (between three Replicat processes) based on the ID column of the source acct table.
(Replicat group 1 parameter file)

```
MAP sales.acct, TARGET sales.acct, FILTER (@RANGE (1, 3, ID));
```

(Replicat group 2 parameter file)

```
MAP sales.acct, TARGET sales.acct, FILTER (@RANGE (2, 3, ID));
```

(Replicat group 3 parameter file)

```
MAP sales.acct, TARGET sales.acct, FILTER (@RANGE (3, 3, ID));
```

Example 2
In the following example, one Extract process splits the processing load into two trails. Since no columns were defined on which to base the range calculation, Oracle GoldenGate will use the primary key columns.

```
RMTTRAIL /ggs/dirdat/aa
TABLE fin.account, FILTER (@RANGE (1, 2));
RMTTRAIL /ggs/dirdat/bb
TABLE fin.account, FILTER (@RANGE (2, 2));
```

Example 3
In the following example, two tables have relative operations based on an order_ID column. The order_master table has a key of order_ID, and the order_detail table has a key of order_ID and item_number. Because the key order_ID 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 sales.order_master, TARGET sales.order_master,
FILTER (@RANGE (1, 2, order_ID));
MAP sales.order_detail, TARGET sales.order_detail,
FILTER (@RANGE (1, 2, order_ID));
```

(Parameter file #2)

```
MAP sales.order_master, TARGET sales.order_master,
FILTER (@RANGE (2, 2, order_ID));
MAP sales.order_detail, TARGET sales.order_detail,
FILTER (@RANGE (2, 2, order_ID));
```

3.3 AFTER

Use the @AFTER function to return the after image of the specified source column. This is the default behavior.

Syntax

```
@AFTER (column)
```

column
The name of the source column for which to return the after image.

Example

```
@AFTER (quantity)
```

3.4 BEFORE

Use the @BEFORE function to return the before image of the specified source column.

When using this parameter, use the GETUPDATEBEFORES parameter in the Extract parameter file to capture before images from the transaction record, or use it in the Replicat parameter file to use the before image in a column mapping or filter. If using the Conflict Resolution and Detection (CDR) feature, you can use the GETBEFORECOLS option of TABLE. To use these parameters, the specified column must be present in the transaction log.

If the database only logs values for changed columns, make certain the required column values are available by enabling supplemental logging for those columns. Alternatively, you can use the FETCHCOLS or FETCHCOLSEXCEPT option of the TABLE parameter. The fetch option involves additional processing overhead.

Syntax

```
@BEFORE (column)
```

column
The name of the source column for which to return the before image.

Example

```
@BEFORE (quantity)
```

3.5 BEFOREAFTER

Use the @BEFOREAFTER function to return the before image if available, or otherwise the after image.

When using this parameter, use the GETUPDATEBEFORES parameter in the Extract parameter file to capture before images from the transaction record, or use it in the Replicat parameter file to use the before image in a column mapping or filter. If using the Conflict Resolution and Detection (CDR) feature, you can use the GETBEFORECOLS option of TABLE. To use these parameters, all columns must be present in the transaction log.

If the database only logs values for changed columns, make certain the required column values are available by enabling supplemental logging for those columns. Alternatively, you can use the FETCHCOLS or FETCHCOLSEXCEPT option of the TABLE parameter. The fetch option involves additional processing overhead.

Syntax

```
@BEFOREAFTER (column)
```

column
The name of the source column for which to return the before image, if available, or otherwise the after image.

Example

```
@BEFOREAFTER (quantity)
```

3.6 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, binary data in a character column 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)
```

column
The name of the target column to which the data will be copied.

Example

The following shows how @BINARY can be used to copy the data from the source column ACCT_CREATE_DATE to the target column ACCT_COMPLAINT.

```
ACCT_COMPLAINT =
@IF ( @NUMBIN (ACCT_CREATE_DATE ) < 48633, 'xxxxxx',
@BINARY (ACCT_COMPLAINT))
```

3.7 BINTOHEX

Use the @BINTOHEX function to convert supplied binary data into its hexadecimal equivalent.

Syntax

```
@BINTOHEX (data)
```

data
Can be one of the following:

- The name of the source column that contains the data
- An expression
- A literal string that is enclosed within single quote marks

Example

`@BINTOHEX ('12345')` converts to `3132333435`.

3.8 CASE

Use the `@CASE` function to select a value depending on a series of value tests. There is no limit to the number of cases you can test with `@CASE`. If the number of cases is large, list the most frequently encountered conditions first for the best performance.

For this function, Oracle GoldenGate supports the use of an escape sequence to represent characters in a string column in Unicode or in the native character encoding of the Microsoft Windows, UNIX, and Linux operating systems. The target column must be a SQL Unicode data type if any argument is supplied as Unicode.

This function does not support `NCHAR` or `NVARCHAR` data types.

Syntax

```
@CASE (value, test_value1, test_result1
[, test_value2, test_result2] [, ...]
[, default_result]
```

value
A value to test, for example, a column name. Enclose literals within single quote marks.

test_value
A valid result for *value*. Enclose literals within single quote marks.

test_result
A value to return based on the value of *test_value*. Enclose literals within single quote marks.

default_result
A default value to return if *value* results in none of the *test_value* values. Enclose literals within single quote marks.

Examples

Example 1
The following returns `A car` if `PRODUCT_CODE` is `CAR` and `A truck` if `PRODUCT_CODE` is `TRUCK`. If `PRODUCT_CODE` fits neither of the first two cases, a `FIELD_MISSING` indication is returned because a default value was not specified.

```
@CASE (PRODUCT_CODE, 'CAR', 'A car', 'TRUCK', 'A truck')
```

Example 2
The following is similar to the previous example, except that it provides for a default value. If PRODUCT_CODE is neither CAR nor TRUCK, the function returns A vehicle.

```
@CASE (PRODUCT_CODE, 'CAR', 'A car', 'TRUCK', 'A truck', 'A vehicle')
```

3.9 COLSTAT

Use the @COLSTAT function to return an indicator to Extract or Replicat that a column is missing, null, or invalid. The indicator can be used as part of a larger manipulation formula that uses additional conversion functions.

Syntax

```
@COLSTAT ({MISSING | NULL | INVALID})
```

Examples

Example 1
The following example returns a NULL into target column ITEM.

```
ITEM = @COLSTAT (NULL)
```

Example 2
The following @IF calculation uses @COLSTAT to return NULL to the target column if PRICE and QUANTITY are less than zero.

```
ORDER_TOTAL = PRICE * QUANTITY, @IF (PRICE < 0 AND QUANTITY < 0, @COLSTAT(NULL))
```

3.10 COLTEST

Use the @COLTEST function to enable conditional calculations by testing for one or more column conditions. If a condition is satisfied, @COLTEST returns TRUE. To perform the conditional calculation, use the @IF function.

Syntax

```
@COLTEST (source_column, test_condition [, test_condition] [, ...])
```

source_column
The name of a source column.

test_condition
Valid values:

PRESENT
Indicates a column is present in the source record and not NULL. Column values can be missing if the database does not log values for columns that do not change, but that is not the same as NULL.

NULL
Indicates a column is present in the source record and NULL.

MISSING
Indicates a column is not present in the source record.

INVALID
Indicates a column is present in the source record but contains invalid data.

Examples

Example 1
The following example uses `@IF` to map a value to the `HIGH_SALARY` column only if the `BASE_SALARY` column in the source record was both present (and not `NULL`) and greater than 250000. Otherwise, `NULL` is returned.

```
HIGH_SALARY =
@IF (@COLTEST (BASE_SALARY, PRESENT) AND
BASE_SALARY > 250000,
BASE_SALARY, @COLSTAT (NULL))
```

Example 2
In the following example, 0 is returned when the `AMT` column is missing or invalid; otherwise a value for `AMT` is returned.

```
AMOUNT = @IF (@COLTEST (AMT, MISSING, INVALID), 0, AMT)
```

3.11 COMPUTE

Use the `@COMPUTE` function to return 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 the `@COMPUTE` phrase when returning the value of an arithmetic expression to another Oracle GoldenGate function, as in:

```
@STRNUM ((AMOUNT1 + AMOUNT2), LEFT)
```

The preceding returns the same result as:

```
@STRNUM ((@COMPUTE (AMOUNT1 + AMOUNT2), LEFT)
```

Arithmetic expressions can be combinations of the following elements.

- Numbers
- The names of columns that contain numbers
- Functions that return numbers
- Arithmetic operators:

  \+ (plus)

  \- (minus)

  * (multiply)

  / (divide)

  \ (remainder)
- Comparison operators:

  > (greater than)

  >= (greater than or equal)

  < (less than)

  <= (less than or equal)

= (equal)

<> (not equal)

Results that are derived from comparisons can be zero (indicating FALSE) or non-zero (indicating TRUE).

- Parentheses (for grouping results in the expression)
- The conjunction operators AND, OR. Oracle GoldenGate only evaluates the necessary part of a conjunction expression. Once a statement is FALSE, the rest of the expression is ignored. This can be valuable when evaluating fields that may be missing or null. For example, if the value of COL1 is 25 and the value of COL2 is 10, then the following are possible:

  @COMPUTE (COL1 > 0 AND COL2 < 3) returns 0.

  @COMPUTE (COL1 < 0 AND COL2 < 3) returns 0. COL2 < 3 is never evaluated.

  @COMPUTE ((COL1 + COL2)/5) returns 7.

Syntax

```
@COMPUTE (expression)
```

expression
A valid arithmetic expression. The numeric value plus the precision cannot be greater than 17 digits. If this limit is exceeded, @COMPUTE returns an error similar to the following.

```
2013-08-01 01:54:22  ERROR   OGG-01334  Error mapping data from column to
column in function COMPUTE.
```

Examples

Example 1

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

Example 2

```
AMOUNT_TOTAL = @IF (AMT >= 0, AMT * 100, 0)
```

Example 3

```
ANNUAL_SALARY = @COMPUTE (MONTHLY_SALARY * 12)
```

```
mod(id,10) = 1 MAP otest.tab,TARGET mtest.tab1,FILTER (@compute (id \ 10)
= 1)
```

This example illustrates how to achieve the remainder SQL. Ensure that there is a space between id \ 10 in the three characters, otherwise it will be reported as a filter syntax error.

3.12 DATE

Use the @DATE function to return dates and times in a variety of formats to the target column based on the format passed into the source column. @DATE converts virtually any type of input into a valid SQL date. @DATE also can be used to extract portions of a date column or to compute a numeric timestamp column based on a date.

Syntax

```
@DATE ('output_descriptor', 'input_descriptor', source_column
[, 'input_descriptor', source_column] [, ...])
```

'output_descriptor'
The output of the function. The valid value is a string that is composed of date descriptors and optional literal values, such as spaces or colons, that are required by the target column. Date descriptors can be strung together as needed. See Table 3-1 for descriptions of date descriptors. The format descriptor must match the `date/time/timestamp` format for the target. Oracle GoldenGate overrides the specified format to make it correct, if necessary.

'input_descriptor'
The source input. The valid value is a string that is composed of date descriptors and optional literal values, such as spaces or colons. Date descriptors can be strung together as needed. The following are examples:

- Descriptor string `'YYYYMMDD'` indicates that the source column specified with *`source_column`* contains (in order) a four-digit year (`YYYY`), month (`MM`), and day (`DD`).
- Descriptor string `'DD/MM/YY'` indicates that the source column specified with *`source_column`* contains the day, a slash, the month, a slash, and the two digit year.

See Table 3-1 for date descriptions.

source_column
The name of the numeric or character source column that supplies the input specified with *`input_descriptor`*.

Table 3-1 Date Descriptors

Descriptor	Description	Valid for...
`CC`	Century	Input/Output
`YY`	Two-digit year	Input/Output
`YYYY`	Four-digit year	Input/Output
`MM`	Numeric month	Input/Output
`MMM`	Alphanumeric month, such as `APR`, `OCT`	Input/Output
`DD`	Numeric day of month	Input/Output
`DDD`	Numeric day of the year, such as `001` or `365`	Input/Output
`DOW0`	Numeric day of the week (Sunday = `0`)	Input/Output

Table 3-1 (Cont.) Date Descriptors

Descriptor	Description	Valid for...
DOW1	Numeric day of the week (Sunday = 1)	Input/Output
DOWA	Alphanumeric day of the week, such as SUN, MON, TUE	Input/Output
HH	Hour	Input/Output
MI	Minute	Input/Output
SS	Seconds	Input/Output
JTSLCT	Use for a Julian timestamp that is already local time, or to keep local time when converting to a Julian timestamp.	Input/Output
JTSGMT	Julian timestamp, the same as JTS.	Input/Output
JTS	Julian timestamp. JUL and JTS produce numbers you can use in numeric expressions. The unit is microseconds. On a Windows machine, the value will be padded with zeros (0) because the granularity of the Windows timestamp is milliseconds.	Input/Output
JUL	Julian day. JUL and JTS produce numbers you can use in numeric expressions.	Input/Output
TTS	NonStop 48-bit timestamp	Input
PHAMIS	PHAMIS application date format	Input
FFFFFF	Fraction (up to microseconds)	Input/Output
STRATUS	STRATUS application timestamp	Input/Output
CDATE	C timestamp in seconds since the Epoch	Input/Output

Examples

Example 1

In an instance where a two-digit year is supplied, but a four-digit year is required in the output, several options exist to obtain the correct century.

- The century can be hard coded, as in:

  ```
  'CC', 19 or 'CC', 20
  ```

- The @IF function can be used to set a condition, as in:

```
'CC', @IF (YY > 70, 19, 20)
```

This causes the century to be set to 19 when the year is greater than 70; otherwise the century is set to 20.

- The system can calculate the century automatically. If the year is less than 50, the system calculates a century of 20; otherwise, a century of 19 is calculated.

Example 2
The following converts year, month and day columns into a date.

```
date_col = @DATE ('YYYY-MM-DD', 'YY', date1_yy, 'MM', date1_mm, 'DD', date1_dd)
```

Example 3
The following converts a date and time, defaulting seconds to zero.

```
date_col = @DATE ('YYYY-MM-DD HH:MI:00', 'YYMMDD', date1, 'HHMI', time1)
```

Example 4
The following converts a numeric column stored as `YYYYMMDDHHMISS` to a SQL date.

```
datetime_col = @DATE ('YYYY-MM-DD HH:MI:SS', 'YYYYMMDDHHMISS', numeric_date)
```

Example 5
The following converts a numeric column stored as `YYYYMMDDHHMISS` to a Julian timestamp.

```
julian_ts_col = @DATE ('JTS', 'YYYYMMDDHHMISS', numeric_date)
```

Example 6
The following converts a Julian timestamp column to two separate columns: a datetime column in the format `YYYY-MM-DD HH:MI:SS` and a fraction column that 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)
```

Example 7
The following produces the time at which an order is filled. The inner `@DATE` expression changes the `order_taken` column into a Julian timestamp, then adds the `order_minutes` column converted into microseconds to this timestamp. The 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.

```
order_filled = @DATE ('YYYY-MM-DD HH:MI:SS', 'JTS', @DATE ('JTS',
'YYMMDDHHMISS', order_taken) + order_minutes * 60 * 1000000)
```

Example 8
The following does a full calculation of times. It goes from a source date column named `dt` to a target column named `dt5` that is to be converted to the date + 5 hours. The calculation also goes from a source timestamp column named `ts` to a target column named `ts5` that is to be converted to the timestamp + 5 hours.

```
MAP scratch.t4, TARGET scratch.t4_copy,
COLMAP ( USEDEFAULTS,
dt5 = @DATE ('YYYY-MM-DD HH:MI:SS', 'JTS',
@COMPUTE (@DATE ('JTS', 'YYYY-MM-DD HH:MI:SS', dt) + 18000000000 ) ),
ts5 = @DATE ('YYYY-MM-DD HH:MI:SS.FFFFFF', 'JTS',
@COMPUTE ( @DATE ('JTS', 'YYYY-MM-DD HH:MI:SS.FFFFFF', ts) + 18000000000 ) )
) ;
```

3.13 DATEDIFF

Use the @DATEDIFF function to calculate the difference between two dates or datetimes, in days or seconds.

Syntax

```
@DATEDIFF ('difference', 'date', 'date')
```

difference
The difference between the specified dates. Valid values can be:

- DD, which computes the difference in days.
- SS, which computes the difference in seconds.

date
A string within single quote marks, in the format of 'YYYY-MM-DD[*HH:MI[:SS]]', where * can be a colon (:) or a blank space, or the @DATENOW function without quotes to return the current date.

Examples

Example 1
The following calculates the number of days since the beginning of the year 2011.

```
YTD = @DATEDIFF ('DD', '2011-01-01', @DATENOW ())
```

Example 2
The following calculates the numerical day of the year. (@DATEDIFF returns 0 for 2011-01-01):

```
todays_day = @COMPUTE (@DATEDIFF ('DD', '2011-01-01', @DATENOW ()) +1)
```

3.14 DATENOW

Use the @DATENOW function to return 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 Saving Time. @DATENOW takes no arguments.

Syntax

```
@DATENOW ()
```

3.15 DDL

Use the @DDL function to return information about a DDL operation.

Syntax

```
@DDL ({TEXT | OPTYPE | OBJNAME | OBJTYPE | OBJOWNER})
```

OBJNAME
Returns the name of the object that is affected by the DDL.

OBJOWNER
Returns the name of the owner of the object that is affected by the DDL.

OBJTYPE
Returns the type of object that is affected by the DDL, such as `TABLE` or `INDEX`)

OPTYPE
Returns the operation type of the DDL, such as `CREATE` or `ALTER`.

TEXT
Returns the first 200 characters of the text of the DDL statement.

Example

The following example uses the output from `@DDL` in an `EVENTACTIONS` shell command.

```
DDL INCLUDE OBJNAME src.t* &
EVENTACTIONS (SHELL ('echo The DDL text is var1> out.txt ', &
VAR var1 = @DDL (TEXT)));
```

The redirected output file might contain a string like this:

```
The DDL text is CREATE TABLE src.test_tab (col1 int);
```

3.16 EVAL

Use the `@EVAL` function to select a value based on a series of independent tests. There is no limit to the number of conditions you can test. If the number of cases is large, list the most frequently encountered conditions first for best performance.

For this function, Oracle GoldenGate supports the use of an escape sequence to represent characters in a string column in Unicode or in the native character encoding of the Microsoft Windows, UNIX, and Linux operating systems. The target column must be a SQL Unicode data type if any argument is supplied as Unicode.

Syntax

```
@EVAL (condition, result
[condition, result] [, ....]
[, default_result])
```

condition
A conditional test using standard conditional operators. More than one condition can be specified.

result
A value or string to return based on the results of the conditional test. Enclose literals within single quote marks. Specify a result for each condition that is used.

default_result
A default result to return if none of the conditions is satisfied. A default result is optional.

NOT_SUPPORTED
In the following example, if the `AMOUNT` column is greater than 10000, a result of `high amount` is returned. If `AMOUNT` is greater than 5000 (and less than or equal to 10000), a

result of `somewhat high` is returned (unless the prior condition was satisfied). If neither condition is satisfied, a `COLUMN_MISSING` indication is returned because a default result is not specified.

```
AMOUNT_DESC = @EVAL (AMOUNT > 10000, 'high amount', AMOUNT > 5000, 'somewhat
high')
```

NOT_SUPPORTED
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')
```

3.17 GETENV

Use the `@GETENV` function to return information about the Oracle GoldenGate environment. You can use the information as input into the following:

- Stored procedures or queries (with `SQLEXEC`)
- Column maps (with the `COLMAP` option of `TABLE` or `MAP`)
- User tokens (defined with the `TOKENS` option of `TABLE` and mapped to target columns by means of the `@TOKEN` function)
- The `GET_ENV_VALUE` user exit function (see "GET_ENV_VALUE")

> **Note:**
>
> All syntax options must be enclosed within quotes as shown in the syntax descriptions.

Syntax

```
@GETENV (
'LAG' , 'unit' |
'LASTERR' , 'error_info' |
'JULIANTIMESTAMP' |
'JULIANTIMESTAMP_PRECISE' |
'RECSOUTPUT' |
{'STATS'|'DELTASTATS'}, ['TABLE', 'table'], 'statistic' |
'GGENVIRONMENT', 'environment_info' |
'GGFILEHEADER', 'header_info' |
'GGHEADER', 'header_info' |
'RECORD', 'location_info' |
'DBENVIRONMENT', 'database_info'
'TRANSACTION', 'transaction_info' |
'OSVARIABLE', 'variable' |
'TLFKEY', SYSKEY, unique_key
'USERNAME',
'OSUSERNAME',
'MACHINENAME',
'PROGRAMNAME',
'CLIENTIDENTIFIER',
)
```

'LAG' , '*unit*'

Valid for Extract and Replicat.

Use the LAG option of @GETENV to return lag information. Lag is the difference between the time that a record was processed by Extract or Replicat and the timestamp of that record in the data source.

Syntax

```
@GETENV ('LAG', {'SEC'|'MSEC'|'MIN'})
```

'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.

'LASTERR' , '*error_info*'

Valid for Replicat.

Use the LASTERR option of @GETENV to return information about the last failed operation processed by Replicat.

Syntax

```
@GETENV ('LASTERR', {'DBERRNUM'|'DBERRMSG'|'OPTYPE'|'ERRTYPE'})
```

'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. For a list of Oracle GoldenGate operation types, see *Administering Oracle GoldenGate*.

'ERRTYPE'
Returns the type of error. Possible results are:

- DB (for database errors)
- MAP (for errors in mapping)

'JULIANTIMESTAMP' | 'JULIANTIMESTAMP_PRECISE'

Valid for Extract and Replicat.

Use the JULIANTIMESTAMP option of @GETENV to return the current time in Julian format. The unit is microseconds (one millionth of a second). On a Windows machine, the value is padded with zeros (0) because the granularity of the Windows timestamp is milliseconds (one thousandth of a second). For example, the following is a typical column mapping:

```
MAP dbo.tab8451, Target targ.tabjts, COLMAP (USEDEFAULTS, &
JTSS = @GETENV ('JULIANTIMESTAMP')
JTSFFFFFF = @date ('yyyy-mm-dd hh:mi:ss.ffffff', 'JTS', &
@getenv ('JULIANTIMESTAMP') ) )
;
```

Possible values that the JTSS and JTSFFFFFF columns can have are:

```
212096320960773000 2010-12-17:16:42:40.773000
212096321536540000 2010-12-17:16:52:16.540000
212096322856385000 2010-12-17:17:14:16.385000
212096323062919000 2010-12-17:17:17:42.919000
212096380852787000 2010-12-18:09:20:52.787000
```

The last three digits (the microseconds) of the number all contain the padding of 0s .

Optionally, you can use the `'JULIANTIMESTAMP_PRECISE'` option to obtain a timestamp with high precision though this may effect performance.

> **Note:**
>
> Do not use these values for ordering operations.
> Instead use this value: `@COMPUTE(@COMPUTE(@NUMSTR(@GETENV ("RECORD", "FILESEQNO")*100000000000)+@NUMSTR(@GETENV ("RECORD", "FILERBA")))"`

Syntax

```
@GETENV ('JULIANTIMESTAMP')
@GETENV ('JULIANTIMESTAMP_PRECISE')
```

`'RECSOUTPUT'`

Valid for Extract.

Use the `RECSOUTPUT` option of `@GETENV` to retrieve a current count of the number of records that Extract has written to the trail file since the process started. The returned value is not unique to a table or transaction, but instead for the Extract session itself. The count resets to 1 whenever Extract stops and then is started again.

Syntax

```
@GETENV ('RECSOUTPUT')
```

`{'STATS'|'DELTASTATS'}, ['TABLE', 'table'], 'statistic'`

Valid for Extract and Replicat.

Use the `STATS` and `DELTASTATS` options of `@GETENV` to return the number of operations that were processed per table for any or all of the following:

- `INSERT` operations
- `UPDATE` operations
- `DELETE` operations
- `TRUNCATE` operations

- Total DML operations
- Total DDL operations
- Number of conflicts that occurred, if the Conflict Detection and Resolution (CDR) feature is used.
- Number of CDR resolutions that succeeded
- Number of CDR resolutions that failed

Any errors in the processing of this function, such as an unresolved table entry or incorrect syntax, returns a zero (0) for the requested statistics value.

Understanding How Recurring Table Specifications Affect Operation Counts

An Extract that is processing the same source table to multiple output trails returns statistics based on each localized output trail to which the table linked to `@GETENV` is written. For example, if Extract captures 100 inserts for table `ABC` and writes table `ABC` to three trails, the result for the `@GETENV` is 300

```
EXTRACT ABC
...
EXTTRAIL c:\ogg\dirdat\aa;
TABLE TEST.ABC;
EXTTRAIL c:\ogg\dirdat\bb;
TABLE TEST.ABC;
TABLE EMI, TOKENS (TOKEN-CNT = @GETENV ('STATS', 'TABLE', 'ABC', 'DML'));
EXTTRAIL c:\ogg\dirdat\cc;
TABLE TEST.ABC;
```

In the case of an Extract that writes a source table multiple times to a single output trail, or in the case of a Replicat that has multiple `MAP` statements for the same `TARGET` table, the statistics results are based on all matching `TARGET` entries. For example, if Replicat filters 20 rows for `REGION 'WEST`,' 10 rows for `REGION 'EAST`,' 5 rows for `REGION 'NORTH`,' and 2 rows for `REGION 'SOUTH'` (all for table `ABC`) the result of the `@GETENV` is 37.

```
REPLICAT ABC
...
MAP TEST.ABC, TARGET TEST.ABC, FILTER (@STREQ (REGION, 'WEST'));
MAP TEST.ABC, TARGET TEST.ABC, FILTER (@STREQ (REGION, 'EAST'));
MAP TEST.ABC, TARGET TEST.ABC, FILTER (@STREQ (REGION, 'NORTH'));
MAP TEST.ABC, TARGET TEST.ABC, FILTER (@STREQ (REGION, 'SOUTH'));
MAP TEST.EMI, TARGET TEST.EMI, &
    COLMAP (CNT = @GETENV ('STATS', 'TABLE', 'ABC', 'DML'));
```

Capturing Multiple Statistics

You can execute multiple instances of `@GETENV` to get counts for different operation types.

This example returns statistics only for `INSERT` and `UPDATE` operations:

```
REPLICAT TEST
..
..
MAP TEST.ABC, TARGET TEST.ABC, COLMAP (USEDEFAULTS, IU = @COMPUTE (@GETENV &
    ('STATS', 'TABLE', 'ABC', 'DML') - (@GETENV ('STATS', 'TABLE', &
    'ABC', 'DELETE'));
```

This example returns statistics for DDL and `TRUNCATE` operations:

```
REPLICAT TEST2
..
..
MAP TEST.ABC, TARGET TEST.ABC, COLMAP (USEDEFAULTS, DDL = @COMPUTE &
(@GETENV ('STATS', 'DDL') + (@GETENV ('STATS', 'TRUNCATE'));
```

Example Use Case

In the following use case, if all DML from the source is applied successfully to the target, Replicat suspends by means of EVENTACTIONS with SUSPEND, until resumed from GGSCI with SEND REPLICAT with RESUME.

GETENV used in Extract parameter file:

```
TABLE HR1.HR*;
TABLE HR1.STAT, TOKENS ('env_stats' = @GETENV ('STATS', 'TABLE', &
    'HR1.HR*', 'DML'));
```

GETENV used in Replicat parameter file:

```
MAP HR1.HR*, TARGET HR2.*;
MAP HR1.STAT, TARGET HR2.STAT, filter (
    @if (
    @token ('stats') =
    @getenv ('STATS', 'TABLE', 'TSSCAT.TCUSTORD', 'DML'), 1, 0 )
    ),
    eventactions (suspend);
```

Using Statistics in FILTER Clauses

Statistics returned by STATS and DELTASTATS are dynamic values and are incremented after mapping is performed. Therefore, when using CDR statistics in a FILTER clause in each of multiple MAP statements, you need to order the MAP statements in descending order of the statistics values. If the order is not correct, Oracle GoldenGate returns error OGG-01921. For detailed information about this requirement, see Document 1556241.1 in the Knowledge base of My Oracle Support at http://support.oracle.com.

Example 3-1 MAP statements containing statistics in FILTER clauses

In the following example, the MAP statements containing the filter for the CDR_CONFLICTS statistic are ordered in descending order of the statistic: >3, then =3, then <3.

```
MAP TEST.GG_HEARTBEAT_TABLE, TARGET TEST.GG_HEARTBEAT_TABLE COMPARECOLS
(ON UPDATE ALL),RESOLVECONFLICT(UPDATEROWEXISTS,(DEFAULT, OVERWRITE)),FILTER
(@GETENV ("STATS", "CDR_CONFLICTS") > 3),EVENTACTIONS (LOG INFO);MAP
TEST.GG_HEARTBEAT_TABLE, TARGET TEST.GG_HEARTBEAT_TABLE COMPARECOLS (ON
UPDATE ALL),RESOLVECONFLICT(UPDATEROWEXISTS,(DEFAULT, OVERWRITE)),FILTER
(@GETENV ("STATS", "CDR_CONFLICTS") = 3),EVENTACTIONS (LOG WARNING);MAP
TEST.GG_HEARTBEAT_TABLE, TARGET TEST.GG_HEARTBEAT_TABLE COMPARECOLS (ON UPDATE
ALL),RESOLVECONFLICT(UPDATEROWEXISTS,(DEFAULT, OVERWRITE)),FILTER (@GETENV
("STATS", "CDR_CONFLICTS") < 3),EVENTACTIONS (LOG WARNING);
```

Syntax

```
@GETENV ({'STATS' | 'DELTASTATS'}, ['TABLE', 'table'], 'statistic')
```

{'STATS' | 'DELTASTATS'}
STATS returns counts since process startup, whereas DELTASTATS returns counts since the last execution of a DELTASTATS.
The execution logic is as follows:

- When Extract processes a transaction record that satisfies @GETENV with STATS or DELTASTATS, the table name is matched against resolved source tables in the TABLE statement.
- When Replicat processes a trail record that satisfies @GETENV with STATS or DELTASTATS, the table name is matched against resolved target tables in the TARGET clause of the MAP statement.

'TABLE', '*table*'
Executes the STATS or DELTASTATS only for the specified table or tables. Without this option, counts are returned for all tables that are specified in TABLE (Extract) or MAP (Replicat) parameters in the parameter file.
Valid *table_name* values are:

- '*schema.table*' specifies a table.
- *'table'* specifies a table of the default schema.
- '*schema.**' specifies all tables of a schema.
- '*' specifies all tables of the default schema.

For example, the following counts DML operations only for tables in the hr schema:

```
MAP fin.*, TARGET fin.*;
MAP hr.*, TARGET hr.*;
MAP hq.rpt, TARGET hq.rpt, COLMAP (USEDEFAULTS, CNT = @GETENV ('STATS',
'TABLE', 'hr.*', 'DML'));
```

Likewise, the following counts DML operations only for the emp table in the hr schema:

```
MAP fin.*, TARGET fin.*;
MAP hr.*, TARGET hr.*;
MAP hq.rpt, TARGET hq.rpt, COLMAP (USEDEFAULTS, CNT = @GETENV ('STATS',
'TABLE', 'hr.emp', 'DML'));
```

By contrast, because there are no specific tables specified for STATS in the following example, the function counts all INSERT, UPDATE, and DELETE operations for all tables in all schemas that are represented in the TARGET clauses of MAP statements:

```
MAP fin.*, TARGET fin.*;
MAP hr.*, TARGET hr.*;
MAP hq.rpt, TARGET hq.rpt, COLMAP (USEDEFAULTS, CNT = &
@GETENV ('STATS', 'DML'));
```

'statistic'
The type of statistic to return. See Using Statistics in FILTER Clauses for important information when using statistics in FILTER clauses in multiple TABLE or MAP statements.

'INSERT'
Returns the number of INSERT operations that were processed.

'UPDATE'
Returns the number of UPDATE operations that were processed.

'DELETE'
Returns the number of DELETE operations that were processed.

'DML'
Returns the total of INSERT, UPDATE, and DELETE operations that were processed.

'TRUNCATE'
Returns the number of TRUNCATE operations that were processed. This variable returns a count only if Oracle GoldenGate DDL replication is not being used. If DDL replication is being used, this variable returns a zero.

'DDL'
Returns the number of DDL operations that were processed, including TRUNCATEs and DDL specified in INCLUDE and EXCLUDE clauses of the DDL parameter, all scopes (MAPPED, UNMAPPED, OTHER). This variable returns a count only if Oracle GoldenGate DDL replication is being used. This variable is not valid for 'DELTASTATS'.

'CDR_CONFLICTS'
Returns the number of conflicts that Replicat detected when executing the Conflict Detection and Resolution (CDR) feature.
Example for a specific table:

```
@GETENV ('STATS','TABLE','HR.EMP','CDR_CONFLICTS')
```

Example for all tables processed by Replicat:

```
@GETENV ('STATS','CDR_CONFLICTS')
```

'CDR_RESOLUTIONS_SUCCEEDED'
Returns the number of conflicts that Replicat resolved when executing the Conflict Detection and Resolution (CDR) feature.
Example for a specific table:

```
@GETENV ('STATS','TABLE','HR.EMP', 'CDR_RESOLUTIONS_SUCCEEDED')
```

Example for all tables processed by Replicat:

```
@GETENV ('STATS','CDR_RESOLUTIONS_SUCCEEDED')
```

'CDR_RESOLUTIONS_FAILED'
Returns the number of conflicts that Replicat could not resolve when executing the Conflict Detection and Resolution (CDR) feature.
Example for a specific table:

```
@GETENV ('STATS','TABLE','HR.EMP', 'CDR_RESOLUTIONS_FAILED')
```

Example for all tables processed by Replicat:

```
@GETENV ('STATS','CDR_RESOLUTIONS_FAILED')
```

'GGENVIRONMENT' , '*environment_info*'

Valid for Extract and Replicat.

Use the GGENVIRONMENT option of @GETENV to return information about the Oracle GoldenGate environment.

Syntax

```
@GETENV ('GGENVIRONMENT', {'DOMAINNAME'|'GROUPDESCRIPTION'|'GROUPNAME'|
   'GROUPTYPE'|'HOSTNAME'|'OSUSERNAME'|'PROCESSID')
```

`'DOMAINNAME'`
(Windows only) Returns the domain name associated with the user that started the process.

`'GROUPDESCRIPTION'`
Returns the description of the group, taken from the checkpoint file. Requires that a description was provided with the `DESCRIPTION` parameter when the group was created with the `ADD` command in GGSCI.

`'GROUPNAME'`
Returns the name of the process group.

`'GROUPTYPE'`
Returns the type of process, either `EXTRACT` or `REPLICAT`.

`'HOSTNAME'`
Returns the name of the system running the Extract or Replicat process.

`'OSUSERNAME'`
Returns the operating system user name that started the process.

`'PROCESSID'`
Returns the process ID that is assigned to the process by the operating system.

`'GGHEADER' , '`*`header_info`*`'`

Valid for Extract and Replicat.

Use the `GGHEADER` option of `@GETENV` to return information from the header portion of an Oracle GoldenGate trail record. The header describes the transaction environment of the record. For more information on record headers and record types, see *Administering Oracle GoldenGate*.

Syntax

```
@GETENV ('GGHEADER', {'BEFOREAFTERINDICATOR'|'COMMITTIMESTAMP'|'LOGPOSITION'|
   'LOGRBA'|'OBJECTNAME'|'TABLENAME'|'OPTYPE'|'RECORDLENGTH'|
   'TRANSACTIONINDICATOR'})
```

> **Note:**
>
> Do not use `TIMESTAMP_PRECISE` for ordering operations.
> Instead use this value: `@COMPUTE(@COMPUTE(@NUMSTR(@GETENV ("RECORD", "FILESEQNO"))*100000000000)+@NUMSTR(@GETENV ("RECORD", "FILERBA")))`

`'BEFOREAFTERINDICATOR'`
Returns the before or after indicator showing whether the record is a before image or an after image. Possible results are:

- BEFORE (before image)
- AFTER (after image)

'COMMITTIMESTAMP'
Returns the transaction timestamp (the time when the transaction committed) expressed in the format of YYYY-MM-DD HH:MI:SS.FFFFFF, for example:

```
2011-01-24 17:08:59.000000
```

'LOGPOSITION'
Returns the position of the Extract process in the data source. (See the LOGRBA option.)

'LOGRBA'
LOGRBA and LOGPOSITION store details of the position in the data source of the record. For transactional log-based products, LOGRBA is the sequence number and LOGPOSITION is the relative byte address. However, these values will vary depending on the capture method and database type.

'OBJECTNAME' | 'TABLENAME'
Returns the table name or object name (if a non-table object).

'OPTYPE'
Returns the type of operation. Possible results are:

```
INSERT
UPDATE
DELETE
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.

'RECORDLENGTH'
Returns the record length in bytes.

'TRANSACTIONINDICATOR'
Returns the transaction indicator. The value corresponds to the TransInd field of the record header, which can be viewed with the Logdump utility.
Possible results are:

- BEGIN (represents TransInD of 0, the first record of a transaction.)
- MIDDLE (represents TransInD of 1, a record in the middle of a transaction.)
- END (represents TransInD of 2, the last record of a transaction.)
- WHOLE (represents TransInD of 3, the only record in a transaction.)

'GGFILEHEADER' , '*header_info*'

Valid for Replicat only.

Use the GGFILEHEADER option of @GETENV to return attributes of an Oracle GoldenGate Extract file or trail file. These attributes are stored as tokens in the file header.

> **Note:**
>
> If a given database, operating system, or Oracle GoldenGate version does not provide information that relates to a given token, a `NULL` value will be returned.

Syntax

```
@GETENV ('GGFILEHEADER', {'COMPATIBILITY'|'CHARSET'|'CREATETIMESTAMP'|
   'FILENAME'|'FILETYPE'|'FILESEQNO'|'FILESIZE'|'FIRSTRECCSN'|
   'LASTRECCSN'|'FIRSTRECIOTIME'|'LASTRECIOTIME'|'URI'|'URIHISTORY'|
   'GROUPNAME'|'DATASOURCE'|'GGMAJORVERSION'|'GGMINORVERSION'|
   'GGVERSIONSTRING'|'GGMAINTENANCELEVEL'|'GGBUGFIXLEVEL'|'GGBUILDNUMBER'|
   'HOSTNAME'|'OSVERSION'|'OSRELEASE'|'OSTYPE'|'HARDWARETYPE'|
   'DBNAME'|'DBINSTANCE'|'DBTYPE'|'DBCHARSET'|'DBMAJORVERSION'|
   'DBMINORVERSION'|'DBVERSIONSTRING'|'DBCLIENTCHARSET'|'DBCLIENTVERSIONSTRING'|
   'LASTCOMPLETECSN'|'LASTCOMPLETEXIDS'|'LASTCSN'|'LASTXID'|
   'LASTCSNTS'|'RECOVERYMODE'})
```

`'COMPATIBILITY'`
Returns the compatibility level of the trail file. The compatibility level of the current Oracle GoldenGate version must be greater than, or equal to, the compatibility level of the trail file to be able to read the data records in that file. Current valid values are from 0 or 6.

- 1 means that the trail file is of Oracle GoldenGate version 10.0 or later, which supports file headers that contain file versioning information.
- 0 means that the trail file is of an Oracle GoldenGate version that is older than 10.0. File headers are not supported in those releases. The 0 value is used for backward compatibility to those Oracle GoldenGate versions.
- 5 means that the trail file is of Oracle GoldenGate version 12.2 or later.
- 6 means that the trail file is of Oracle GoldenGate version 12.3.0.1.

  This value keeps increasing as per the Oracle GoldenGate version depending on the trail file version.

`'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.

`'FILENAME'`
Returns the name of the trail file. Can be an absolute or relative path, with a forward or backward slash depending on the file system.

`'FILETYPE'`
Returns a numerical value 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. The valid values are:

- 0 - EXTFILE

- 1 - EXTTRAIL
- 2 - UNIFIED and EXTFILE
- 3 - UNIFIED and EXTTRAIL

'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. For more information about the CSN, see *Administering Oracle GoldenGate*.

'LASTRECCSN'
Returns the commit sequence number (CSN) of the last record in the trail file.Value is `NULL` until the trail file is completed. For more information about the CSN, see *Administering Oracle GoldenGate*.

'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.

'RECOVERYMODE'
Returns recovery information for internal Oracle GoldenGate use. It is usually set to `APPENDMODE`.

'URI'
Returns the universal resource identifier of the process that created the trail file, in the following format:

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 Oracle GoldenGate installation path.
- `group_name` is the name of the process group that is linked with the process.

The following example shows where the trail was processed and by which process. This includes a history of previous runs.

```
sys1:home:oracle:v9.5:extora
```

'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.

`'GROUPNAME'`
Returns the name of the group that is associated with the Extract process that created the trail. The group name is the one that was supplied when the `ADD EXTRACT` command was issued.

`'DATASOURCE'`
Returns the data source that was read by the process as a number. The return value can be one of the following:

- `DS_EXTRACT_TRAILS`: The source was an Oracle GoldenGate extract file, populated with change data. The return value is `0`.
- `DS_DATABASE:` The source was a direct select from database table written to a trail, used for `SOURCEISTABLE`-driven initial load. The return value is `2`.
- `DS_TRAN_LOGS`: The source was the database transaction log. The return value is `3`.
- `DS_INITIAL_DATA_LOAD`: The source was a direct select from database tables for an initial load. The return value is `4`.
- `DS_VAM_EXTRACT:` The source was a vendor access module (VAM). The return value is `5`.
- `DS_VAM_TWO_PHASE_COMMIT:` The source was a VAM trail. The return value is `6`.

`'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.

`'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.

`'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:

- Version s10_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

'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

'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.

'DBTYPE'
Returns the type of database that produced the data in the trail file. Can be one of:

```
DB2 UDB
DB2 ZOS
MSSQL
MYSQL
ORACLE
TERADATA
ODBC
```

'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.)

'LASTCOMPLETECSN'
Returns recovery information for internal Oracle GoldenGate use.

'LASTCOMPLETEXIDS'
Returns recovery information for internal Oracle GoldenGate use.

'LASTCSN'
Returns recovery information for internal Oracle GoldenGate use.

'LASTXID'
Returns recovery information for internal Oracle GoldenGate use.

'LASTCSNTS'
Returns recovery information for internal Oracle GoldenGate use.

'RECORD' , '*location_info*'

Valid for a data pump Extract or Replicat.

Use the `RECORD` option of `@GETENV` to return the location or Oracle rowid of a record in an Oracle GoldenGate trail file.

Syntax

```
@GETENV ('RECORD',
{'TIMESTAMP_PRECISE'|'FILESEQNO'|'FILERBA'|'ROWID'|'RSN'|'TIMESTAMP'})
```

'TIMESTAMP_PRECISE'
Valid for a data pump, Extract, or Replicat.
The `TIMESTAMP_PRECISE` option returns the timestamp from year to microseconds. However, depending on the database, the value can be in milliseconds with 0 microseconds.

'FILESEQNO'
Returns the sequence number of the trail file without any leading zeros.

'FILERBA'
Returns the relative byte address of the record within the FILESEQNO file.

'ROWID'
(Valid for Oracle) Returns the row id of the record.

'RSN'
Returns the record sequence number within the transaction.

'TIMESTAMP'
Returns the timestamp of the record.

Example:

```
REC-TIMESTAMP: 2017-10-31 06:21:07 REC-TIMESTAMP-PRECISE: 2017-10-31
06:21:07.478064
```

'DBENVIRONMENT' , '*database_info*'

Valid for Extract and Replicat.

Use the DBENVIRONMENT option of @GETENV to return global environment information for a database.

Syntax

```
@GETENV ('DBENVIRONMENT', {'DBNAME'|'DBVERSION'|'DBUSER'|'SERVERNAME'})
```

'DBNAME'
Returns the database name.

'DBVERSION'
Returns the database version.

'DBUSER'
Returns the database login user. Note that SQL Server does not log the user ID.

'SERVERNAME'
Returns the name of the server.

'TRANSACTION' , '*transaction_info*

Valid for Extract.

Use the TRANSACTION option of @GETENV to return information about a source transaction. This option is valid for the Extract process but not for pump Extract and Replicat.

Syntax

```
@GETENV ('TRANSACTION',
{'TIMESTAMP_PRECISE'|'TRANSACTIONID'|'XID'|'CSN'|'TIMESTAMP'|'NAME'|
   'USERID'|'USERNAME'|'PLANNAME' | 'LOGBSN' | 'REDOTHREAD' | 'PROGRAMNAME' |
'CLIENTIDENTIFIER' | 'MACHINENAME' | 'USERNAME')
```

> **Note:**
>
> Do not use TIMETSAMP_PRECISE or TIMESTAMP for ordering operations. Instead use this value: @COMPUTE(@COMPUTE(@NUMSTR(@GETENV ("RECORD", "FILESEQNO"))*100000000000)+@NUMSTR(@GETENV ("RECORD", "FILERBA")))

'TIMESTAMP_PRECISE'
This option is valid for Extract. Use the TIMESTAMP_PRECISE returns the timestamp from year to microseconds. However, depending on the database, the value can be in milliseconds with 0 microseconds

'TRANSACTIONID' | 'XID'
Returns the transaction ID number. Either TRANSACTIONID or XID can be used. The transaction ID and the CSN are associated with the first record of every transaction and are stored as tokens in the trail record. For each transaction ID, there is an associated CSN. Transaction ID tokens have no zero-padding on any platform, because they never get evaluated as relative values. They only get evaluated for whether they match or do not match. Note that in the trail, the transaction ID token is shown as TRANID.

'CSN'
Returns the commit sequence number (CSN). The CSN is not zero-padded when returned for these databases: Oracle, DB2 LUW, and DB2 z/OS. For all other supported databases, the CSN is zero-padded.
Note that in the trail, the CSN token is shown as LOGCSN. See the TRANSACTIONID | XID environment value for additional information about the CSN token.
For more information about the CSN, see *Administering Oracle GoldenGate*.

'TIMESTAMP'
Returns the commit timestamp of the transaction.

'NAME'
Returns the transaction name, if available.

'USERID'
(Oracle) Returns the Oracle user ID of the database user that committed the last transaction. This is not valid for pump Extract and/or Replicat.

'USERNAME'
(Oracle) Returns the Oracle user name of the database user that committed the last transaction. This is not valid for pump Extract and/or Replicat.

'PLANNAME'
(DB2 z/OS) Returns the plan name under which the current transaction was originally executed. The plan name is included in the begin unit of recovery log record.

'LOGBSN'
Returns the begin sequence number (BSN) in the transaction log. The BSN is the native sequence number that identifies the beginning of the oldest uncommitted transaction that is held in Extract memory. For example, given an Oracle database, the BSN would be expressed as a system change number (SCN). The BSN corresponds to the current I/O checkpoint value of Extract. This value can be obtained

from the trail by Replicat when @GETENV ('TRANSACTION', 'LOGBSN') is used. This value also can be obtained by using the INFO REPLICAT command with the DETAIL option. The purpose of obtaining the BSN from Replicat is to get a recovery point for Extract in the event that a system failure or file system corruption makes the Extract checkpoint file unusable. See *Administering Oracle GoldenGate* for more information about recovering the Extract position.

'REDOTHREAD'
Returns the thread number of a RAC node extract; on non-RAC node extracts the value is always 1. For data pump and Replicat, the thread id used by Extract capture of a RAC node is returned; on non-RAC, @GETENV() returns an error. Logdump shows the token, ORATHREADID, in the token section if the transaction is captured by Extract on a RAC node.

'PROGRAMNAME'
Name of the program or application that started the transaction or session.

'CLIENTIDENTIFIER'
Value set by using DBMS_SESSION_.set_identifier().

'MACHINENAME'
Name of the host, machine, or server where database is running

'USERNAME'
Database login user name.

Example:

```
DB2 zOS:
TRANS-TIMESTAMP:          2017-10-31 06:21:07
TRANS-TIMESTAMP-PRECISE: 2017-10-31 06:21:07.485792
```

'OSVARIABLE' , '*variable*'

Valid for Extract and Replicat.

Use the OSVARIABLE option of @GETENV to return the string value of a specified operating-system environment variable.

Syntax

```
@GETENV ('OSVARIABLE', 'variable')
```

'variable'
The name of the variable. The search is an exact match of the supplied variable name. For example, the UNIX grep command would return all of the following variables, but @GETENV ('OSVARIABLE', 'HOME') would only return the value for HOME:

```
ANT_HOME=/usr/local/ant
JAVA_HOME=/usr/java/j2sdk1.4.2_10
HOME=/home/judyd
ORACLE_HOME=/rdbms/oracle/ora1022i/64
```

The search is case-sensitive if the operating system supports case-sensitivity.

'TLFKEY' , SYSKEY, '*unique_key*'

Valid for Extract and Replicat.

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).

Syntax

```
@GETENV ('TLFKEY', SYSKEY, unique_key)
```

SYSKEY, *unique_key*
The NonStop node number of the source TLF/PTLF file. Do not enclose this syntax element in quotes.
Example:

```
GETENV ('TLFKEY', SYSKEY, 27)
```

3.18 GETVAL

Use the @GETVAL function to extract values from a stored procedure or 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 stored procedure or query executed successfully.
2. Whether or not the stored procedure or query results have expired.

When a value cannot be extracted, the @GETVAL function results in a "column missing" condition. Typically, this occurs for update operations if the database only logs values for columns that were changed.

Usually this means that the column cannot be mapped. To test for missing column values, use the @COLTEST function to test the result of @GETVAL, and then map an alternative value for the column to compensate for missing values, if desired. Or, to ensure that column values are available, you can use the FETCHCOLS or FETCHCOLSEXCEPT option of the TABLE or MAP parameter to fetch the values from the database if they are not present in the log. Enabling supplemental logging for the necessary columns also would work.

Syntax

```
@GETVAL (name.parameter)
```

name
The name of the stored procedure or query. When using SQLEXEC to execute the procedure or query, valid values are as follows:
For queries, use the logical name specified with the ID option of the SQLEXEC clause. ID is a required SQLEXEC argument for queries.
For stored procedures, use one of the following, depending on how many times the procedure is to be executed within a TABLE or MAP statement:

- For multiple executions, use the logical name defined by the `ID` clause of the `SQLEXEC` statement. `ID` is required for multiple executions of a procedure.
- For a single execution, use the actual stored procedure name.

parameter
Valid values are one of the following.

- The name of the parameter in the stored procedure or query from which the data will be extracted and passed to the column map.
- `RETURN_VALUE`, if extracting values returned by a stored procedure or query.

Alternate Syntax

With `SQLEXEC`, you can capture parameter results without explicitly using the `@GETVAL` keyword. Simply refer to the procedure name (or logical name if using a query or multiple instances of a procedure) and parameter in the following format:

```
{procedure_name | logical_name}.parameter
```

Examples, Standard Syntax

Example 1
The following enables each map statement to call the stored procedure `lookup` by referencing the logical names `lookup1` and `lookup2` within the `@GETVAL` function and refer appropriately to each set of results.

```
MAP schema.srctab, TARGET schema.targtab,
SQLEXEC (SPNAME lookup, ID lookup1, PARAMS (param1 = srccol)),
COLMAP (targcol1 = @GETVAL (lookup1.param2));
MAP schema.srctab, TARGET schema.targtab2,
SQLEXEC (SPNAME lookup, ID lookup2, PARAMS (param1 = srccol)),
COLMAP (targcol2= @GETVAL (lookup2.param2));
```

Example 2
The following shows a single execution of the stored procedure `lookup`. In this case, the actual name of the procedure is used. A logical name is not needed.

```
MAP schema.tab1, TARGET schema.tab2,
SQLEXEC (SPNAME lookup, PARAMS (param1 = srccol)),
COLMAP (targcol = @GETVAL (lookup.param1));
```

Example 3
The following shows the execution of a query from which values are mapped with `@GETVAL`.

```
MAP sales.account, TARGET sales.newacct,
SQLEXEC (ID lookup,
QUERY ' select desc_col into desc_param from lookup_table '
' where code_col = :code_param ',
PARAMS (code_param = account_code)),
COLMAP (newacct_id = account_id, newacct_val = @GETVAL (lookup.desc_param));
```

Examples, Alternate Syntax

Example 1
In the following example, `@GETVAL` is called implicitly for the phrase `proc1.p2` without the `@GETVAL` keyword.

```
MAP test.tab1, TARGET test.tab2,
SQLEXEC (SPNAME proc1, ID myproc, PARAMS (p1 = sourcecol1)),
COLMAP (targcol1 = proc1.p2);
```

Example 2
In the following example, the `@GETVAL` function is called implicitly for the phrase `lookup.desc_param` without the `@GETVAL` keyword.

```
MAP sales.account, TARGET sales.newacct,
SQLEXEC (ID lookup,
QUERY ' select desc_col into desc_param from lookup_table '
' where code_col = :code_param ',
PARAMS (code_param = account_code)),
COLMAP (newacct_id = account_id, newacct_val = lookup.desc_param);
```

3.19 HEXTOBIN

Use the `@HEXTOBIN` function to convert a supplied string of hexadecimal data into raw format.

Syntax

```
@HEXTOBIN (data)
```

data
The name of the source column, an expression, or a literal string that is enclosed within double quote marks.

Example

`@HEXTOBIN ('414243')` converts to three bytes: `0x41 0x42 0x43`.

3.20 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 names.

Use `@HIGHVAL` and `@LOWVAL` only with string and binary data types. When using them with strings, only `@STRNCMP` is valid. Using them with decimal or date data types or with `SQLEXEC` operations can cause errors. `DOUBLE` data types result in `-1` or `0` (Oracle `NUMBER`, no precision, no scale).

Syntax

```
@HIGHVAL ([length]) | @LOWVAL ([length])
```

length
Optional. Specifies the binary output length in bytes. The maximum value of *length* is the length of the target column.

Example

The following example assumes that the size of the `group_level` column is 5 bytes.

Function statement	Result
`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}`

3.21 IF

Use the `@IF` function to return one of two values, based on a condition. You can use the `@IF` function with other Oracle GoldenGate functions to begin a conditional argument that tests for one or more exception conditions. You can direct processing based on the results of the test. You can nest `@IF` statements, if needed.

Syntax

`@IF (`*`condition`*`, `*`value_if_non-zero`*`, `*`value_if-zero`*`)`

`condition`
A valid conditional expression or Oracle GoldenGate function. Use numeric operators (such as =, > or <) only for numeric comparisons. For character comparisons, use one of the character-comparison functions.

`value_if_non-zero`
Non-zero is considered `true`.

`value_if_zero`
Zero (0) is considered `false`.

Examples

Example 1
The following returns an amount only if the `AMT` column is greater than zero; otherwise zero is returned.

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

Example 2
The following returns `WEST` if the `STATE` column is `CA`, `AZ` or `NV`; otherwise it returns `EAST`.

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

Example 3
The following returns the result of the `PRICE` column multiplied by the `QUANTITY` column if both columns are greater than 0. Otherwise, the `@COLSTAT (NULL)` function creates a `NULL` value in the target column.

```
ORDER_TOTAL = @IF (PRICE > 0 AND QUANTITY > 0, PRICE * QUANTITY,
@COLSTAT (NULL))
```

Example 4
The following example demonstrates a nested @IF statement. In the example, if the QUANTITY is more than 10, then the item price is 90% of thePRICE.

```
 ORDER_TOTAL = @IF (PRICE > 0 AND QUANTITY > 0, @IF (QUANTITY > 10, (PRICE *
0.9) * QUANTITY, PRICE * QUANTITY), @COLSTAT(NULL))
```

> **Note:**
>
> When enclosed in parenthesis (), Oracle GoldenGate column mapping function expects numeric results. The column value must be specified using single quotes.

3.22 NUMBIN

Use the @NUMBIN function to convert a binary string of eight or fewer bytes into a number. Use this function when the source column defines a byte stream that actually is a number represented as a string.

Syntax

```
@NUMBIN (source_column)
```

source_column
The name of the source column that contains the string to be converted.

Example

The following combines @NUMBIN and @DATE to transform a 48-bit column to a 64-bit Julian value for local time.

```
DATE = @DATE ('JTSLCT', 'TTS' @NUMBIN (DATE))
```

3.23 NUMSTR

Use the @NUMSTR function to convert a string (character) column or value into a number. Use @NUMSTR to do either of the following:

- Map a string (character) to a number.
- Use a string column that contains only numbers in an arithmetic expression.

Syntax

```
@NUMSTR (input)
```

input
Can be either of the following:

- The name of a character column.
- A literal string that is enclosed within single quote marks.

Example

```
PAGE_NUM = @NUMSTR (ALPHA_PAGE_NO)
```

3.24 OGG_SHA1

Use the OGG_SHA1 function to return the SHA-1 160 bit / 20 bytes hash value.

Syntax

```
OGG_SHA1(expression)
```

expression
The name of a column, literal string, other column mapping function.

Example

```
OGG_SHA1(col_name)
```

3.25 STRCAT

Use the @STRCAT function to concatenate one or more strings or string (character) columns. Enclose literal strings within single quote marks.

For this function, Oracle GoldenGate supports the use of an escape sequence to represent characters in a string column in Unicode or in the native character encoding of the Microsoft Windows, UNIX, and Linux operating systems. The target column must be a SQL Unicode data type if any argument is supplied as Unicode.

Syntax

```
@STRCAT (string1, string2 [, ...])
```

string1
The first column or literal string to be concatenated.

string2
The next column or literal string to be concatenated.

Example

The following creates a phone number from three columns and includes the literal formatting values.

```
PHONE_NO = @STRCAT (AREA_CODE, PREFIX, '-', PHONE)
```

3.26 STRCMP

Use the @STRCMP function to compare two character columns or literal strings. Enclose literals within single quote marks.

@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.

For this function, Oracle GoldenGate supports the use of an escape sequence to represent characters in a string column in Unicode or in the native character encoding of the Microsoft Windows, UNIX, and Linux operating systems.

This function does not support `NCHAR` or `NVARCHAR` data types.

Syntax

```
@STRCMP (string1, string2)
```

string1
The first column or literal string to be compared.

string2
The second column or literal string to be compared.

Example

The following example compares two literal strings and returns `1` because the first string is greater than the second one.

```
@STRCMP ('JOHNSON', 'JONES')
```

3.27 STRCMPNULL

Use the `@STRCMPNULL` in the same way as `@STRCMP` function to compare two character columns or literal strings, but if the arguments are `NULL`, the result value is `0` instead of `NULL`.

Syntax

```
@STRCMPNULL (string1, string2)
```

string1
The first column or literal string to be compared.

string2
The second column or literal string to be compared.

3.28 STREQ

Use the `@STREQ` function to determine whether or not two string (character) columns or literal strings are equal. Enclose literals within single quote marks. `@STREQ` returns the following:

- `1` (true) if the strings are equal.
- `0` (false) if the strings are not equal.

For this function, Oracle GoldenGate supports the use of an escape sequence to represent characters in a string column in Unicode or in the native character encoding of the Microsoft Windows, UNIX, and Linux operating systems.

Trailing spaces are truncated before comparing the strings.

This function does not support NCHAR or NVARCHAR data types.

Syntax

```
@STREQ (string1, string2)
```

string1
The first column or literal string to be compared.

string2
The second column or literal string to be compared.

Example

The following compares the value of the region column to the literal value EAST. If region = EAST, the record passes the filter.

```
FILTER (@STREQ (region, 'EAST'))
```

You could use @STREQ in a comparison to determine a result, as shown in the following example. If the state is NY, the expression returns East Coast. Otherwise, it returns Other.

```
@IF (@STREQ (state, 'NY'), 'East Coast', 'Other')
```

3.29 STREQNULL

Use the @STREQNULL function in the same way as @STREQ to determine whether or not two string (character) columns or literal strings are equal. However, if the two arguments passed to the function are NULL, then the return value is 1.

Syntax

```
@STREQNULL (string1, string2)
```

string1
The first column or literal string to be compared.

string2
The second column or literal string to be compared.

3.30 STREXT

Use the @STREXT function to extract a portion of a string.

For this function, Oracle GoldenGate supports the use of an escape sequence to represent characters in a string column in Unicode or in the native character encoding of the Microsoft Windows, UNIX, and Linux operating systems. The target column must be a SQL Unicode data type if any argument is supplied as Unicode.

This function does not support NCHAR or NVARCHAR data types.

Syntax

```
@STREXT (string, begin_position, end_position)
```

string
The string from which to extract. The string can be either the name of a character column or a literal string. Enclose literals within single quote marks.

begin_position
The character position at which to begin extraction.

end_position
The character position at which to end extraction. 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)
```

3.31 STRFIND

Use the `@STRFIND` function to determine the position of a string within a string column or else return zero if the string is not found. Optionally, `@STRFIND` can accept a starting position within the string.

For this function, Oracle GoldenGate supports the use of an escape sequence to represent characters in a string column in Unicode or in the native character encoding of the Microsoft Windows, UNIX, and Linux operating systems. The target column must be a SQL Unicode data type if any argument is supplied as Unicode.

This function does not support `NCHAR` or `NVARCHAR` data types.

Syntax

```
@STRFIND (string, 'search_string' [,  begin_position])
```

string
The string in which to search. This can be either the name of a character column or a literal string that is within single quote marks.

'search_string'
The string for which to search. Enclose the search string within single quote marks.

begin_position
The byte position at which to begin searching.

Example

Assuming the string for the `ACCT` column is `ABC123ABC`, the following are possible results.

Function statement	Result
`@STRFIND (ACCT, '23')`	5

Function statement	Result
`@STRFIND (ACCT, 'ZZ')`	0
`@STRFIND (ACCT, 'ABC', 2)`	7 (because the search started at the second byte)

3.32 STRLEN

Use the `@STRLEN` function to return the length of a string, expressed as the number of characters.

For this function, Oracle GoldenGate supports the use of an escape sequence to represent characters in a string column in Unicode or in the native character encoding of the Microsoft Windows, UNIX, and Linux operating systems. The target column must be a SQL Unicode data type if any argument is supplied as Unicode.

This function does not support `NCHAR` or `NVARCHAR` data types.

Syntax

```
@STRLEN (string)
```

string
The name of a string (character) column or a literal string. Enclose literals within single quote marks.

Examples

```
@STRLEN (ID_NO)

@STRLEN ('abcd')
```

3.33 STRLTRIM

Use the `@STRLTRIM` function to trim leading spaces.

For this function, Oracle GoldenGate supports the use of an escape sequence to represent characters in a string column in Unicode or in the native character encoding of the Microsoft Windows, UNIX, and Linux operating systems. The target column must be a SQL Unicode data type if any argument is supplied as Unicode.

Syntax

```
@STRLTRIM (string)
```

string
The name of a character column or a literal string that is enclosed within single quote marks.

Example

```
birth_state = @strltrim (state)
```

3.34 STRNCAT

Use the @STRNCAT function to concatenate one or more strings to a maximum length.

For this function, Oracle GoldenGate supports the use of an escape sequence to represent characters in a string column in Unicode or in the native character encoding of the Microsoft Windows, UNIX, and Linux operating systems. The target column must be a SQL Unicode data type if any argument is supplied as Unicode.

This function does not support NCHAR or NVARCHAR data types.

Syntax

```
@STRNCAT (string, max_length [, string, max_length] [, ...] )
```

string
The name of a string (character) column or a literal string that is enclosed within single quote marks.

max_length
The maximum string length, in characters.

Example

The following concatenates two strings and results in ABC123.

```
PHONE_NO = @STRNCAT ('ABCDEF', 3, '123456', 3)
```

3.35 STRNCMP

Use the @STRNCMP function to compare two strings based on a specific number of bytes. The string can be either the name of a string (character) column or a literal string that is enclosed within single quote marks. The comparison starts at the first byte in the string.

@STRNCMP 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.

This function does not support NCHAR or NVARCHAR data types.

Syntax

```
@STRNCMP (string1, string2, max_length)
```

string1
The first string to be compared.

string2
The second string to be compared.

max_length
The maximum number of bytes in the string to compare.

Example

The following example compares the first two bytes of each string, as specified by a *max_length* of 2, and it returns `0` because both sets are the same.

```
@STRNCMP ('JOHNSON', 'JONES', 2)
```

3.36 STRNUM

Use the `@STRNUM` function to convert a number into a string and specify the output format and padding.

Syntax

```
@STRNUM (column, {LEFT | LEFTSPACE, | RIGHT | RIGHTZERO} [length] )
```

column
The name of a source numeric column.

`LEFT`
Left justify, without padding.

`LEFTSPACE`
Left justify, fill the rest of the target column with spaces.

`RIGHT`
Right justify, fill the rest of the target column with spaces. If the value of a column is a negative value, the spaces are added before the minus sign. For example, `strnum(Col1, right)` used for a column value of -1.27 becomes ###-1.27, assuming the target column allows 7 digits. The minus sign is not counted as a digit, but the decimal is.

`RIGHTZERO`
Right justify, fill the rest of the target column with zeros. If the value of a column is a negative value, the zeros are added after the minus sign and before the numbers. For example, `strnum(Col1, rightzero)` used for a column value of -1.27 becomes -0001.27, assuming the target column allows 7 digits. The minus sign is not counted as a digit, but the decimal is.

length
Specifies the output length, when any of the options are used that specify padding (all but `LEFT`). For example:

- `strnum(Col1, right, 6)` used for a column value of -1.27 becomes ##-1.27. The minus sign is not counted as a digit, but the decimal is.
- `strnum(Col1, rightzero, 6)` used for a column value of -1.27 becomes -001.27. The minus sign is not counted as a digit, but the decimal is.

Example

Assuming a source column named `NUM` has a value of 15 and the target column's maximum length is 5 characters, the following examples show the different types of results obtained with formatting options.

Function statement	Result (# denotes a space)
`CHAR1 = @STRNUM (NUM, LEFT)`	`15`
`CHAR1 = @STRNUM (NUM, LEFTSPACE)`	`15###`
`CHAR1 = @STRNUM (NUM, RIGHTZERO)`	`00015`
`CHAR1 = @STRNUM (NUM, RIGHT)`	`###15`

If an output *length* of 4 is specified in the preceding example, the following shows the different types of results.

Function statement	Result (# denotes a space)
`CHAR1 = @STRNUM (NUM, LEFTSPACE, 4)`	`15##`
`CHAR1 = @STRNUM (NUM, RIGHTZERO, 4)`	`0015`
`CHAR1 = @STRNUM (NUM, RIGHT, 4)`	`##15`

3.37 STRRTRIM

Use the `@STRRTRIM` function to trim trailing spaces.

For this function, Oracle GoldenGate supports the use of an escape sequence to represent characters in a string column in Unicode or in the native character encoding of the Microsoft Windows, UNIX, and Linux operating systems. The target column must be a SQL Unicode data type if any argument is supplied as Unicode.

Syntax

```
@STRRTRIM (string)
```

string
The name of a character column or a literal string that is enclosed within single quote marks.

Example

```
street_address = @strrtrim (address)
```

3.38 STRSUB

Use the `@STRSUB` function to substitute strings within a string (character) column or constant. Enclose literal strings within single quote marks.

For this function, Oracle GoldenGate supports the use of an escape sequence to represent characters in a string column in Unicode or in the native character encoding

of the Microsoft Windows, UNIX, and Linux operating systems. The target column must be a SQL Unicode data type if any argument is supplied as Unicode.

Any single byte code value 1 to 255 can be used in hexadecimal or octal format for the string arguments. Hex values A to F are case insensitive and the leading 'x' must be lower case. Value zero (0) (`\x00` and `\000`) is not allowed because it is a string terminator. No multibyte character set value or UNICODE values are supported.

This function does not support `NCHAR` or `NVARCHAR` data types.

Syntax

```
@STRSUB
(source_string, search_string, substitute_string
[, search_string, substitute_string] [, ...])
```

source_string
A source string, within single quotes, or the name of a source column that contains the characters for which substitution is to occur.

search_string
The string, within single quotes, for which substitution is to occur.

substitute_string
The string, within single quotes, that will be substituted for the search string.

Examples

Example 1
The following returns `xxABCxx`.

```
@STRSUB ('123ABC123', '123', 'xx')
```

Example 2
The following returns `023zBC023`.

```
@STRSUB ('123ABC123', 'A', 'z', '1', '0')
```

Example 3
The following is an example of replacing `^Z`, using a hexadecimal string argument, with a space.

```
@strsub (col1,'\x1A',' '));
```

3.39 STRTRIM

Use the `@STRTRIM` function to trim leading and trailing spaces.

For this function, Oracle GoldenGate supports the use of an escape sequence to represent characters in a string column in Unicode or in the native character encoding of the Microsoft Windows, UNIX, and Linux operating systems. The target column must be a SQL Unicode data type if any argument is supplied as Unicode.

Syntax

```
@STRTRIM (string)
```

string
The name of a character column or a literal string that is enclosed within single quote marks.

Example

```
pin_no = @strtrim (custpin)
```

3.40 STRUP

Use the @STRUP function to change an alphanumeric string or string (character) column to upper case.

For this function, Oracle GoldenGate supports the use of an escape sequence to represent characters in a string column in Unicode or in the native character encoding of the Microsoft Windows, UNIX, and Linux operating systems. The target column must be a SQL Unicode data type if any argument is supplied as Unicode.

This function does not support NCHAR or NVARCHAR data types.

Syntax

```
@STRUP (string)
```

string
The name of a character column or a literal string that is enclosed within single quote marks.

Example

The following returns SALESPERSON.

```
@STRUP ('salesperson')
```

3.41 TOKEN

Use the @TOKEN function to retrieve token data that is stored in the user token area of the Oracle GoldenGate record header. You can map token data to a target column by using @TOKEN in the source expression of a COLMAP clause. As an alternative, you can use @TOKEN within a SQLEXEC statement, an Oracle GoldenGate macro, or a user exit.

To define token data, use the TOKENS clause of the TABLE parameter in the Extract parameter file. For more information about using tokens, see Administering Oracle GoldenGate for Windows and UNIX.

Syntax

```
@TOKEN ('token')
```

'token'
The name, enclosed within single quote marks, of the token for which data is to be retrieved.

Example

In the following example, 10 tokens are mapped to target columns.

```
MAP ora.oratest, TARGET ora.rpt,
COLMAP (
host = @token ('tk_host'),
gg_group = @token ('tk_group'),
osuser = @token ('tk_osuser'),
domain = @token ('tk_domain'),
ba_ind = @token ('tk_ba_ind'),
commit_ts = @token ('tk_commit_ts'),
pos = @token ('tk_pos'),
rba = @token ('tk_rba'),
tablename = @token ('tk_table'),
optype = @token ('tk_optype')
);
```

3.42 VALONEOF

Use the `@VALONEOF` function to compare a string or string (character) column to a list of values. If the value or column is in the list, `1` is returned; otherwise `0` is returned. This function trims trailing spaces before the comparison.

For this function, Oracle GoldenGate supports the use of an escape sequence to represent characters in a string column in Unicode or in the native character encoding of the Microsoft Windows, UNIX, and Linux operating systems. The target column must be a SQL Unicode data type if any argument is supplied as Unicode.

This function does not support `NCHAR` or `NVARCHAR` data types.

Syntax

```
@VALONEOF (expression, value [, value] [, ...])
```

expression
The name of a character column or a literal enclosed within single quote marks.

value
A criteria value.

Example

In the following example, if `STATE` is `CA` or `NY`, the expression returns `COAST`, which is the response returned by `@IF` when the value is non-zero (`true`). Otherwise, the expression returns `MIDDLE`.

```
@IF (@VALONEOF (STATE, 'CA', 'NY'), 'COAST', 'MIDDLE')
```

4
User Exit Functions

This chapter describes the Oracle GoldenGate user exit functions and their syntax and includes the following topics:
For more information about using Oracle GoldenGate user exits, see *Administering Oracle GoldenGate*.

Topics:

- Calling a User Exit
- Summary of User Exit Functions
- Using EXIT_CALL_TYPE
- Using EXIT_CALL_RESULT
- Using EXIT_PARAMS
- Using ERCALLBACK
- Function Codes
- COMPRESS_RECORD
- DECOMPRESS_RECORD
- GET_BASE_OBJECT_NAME
- GET_BASE_OBJECT_NAME_ONLY
- GET_BASE_SCHEMA_NAME_ONLY
- GET_BEFORE_AFTER_IND
- GET_CATALOG_NAME_ONLY
- GET_COL_METADATA_FROM_INDEX
- GET_COL_METADATA_FROM_NAME
- GET_COLUMN_INDEX_FROM_NAME
- GET_COLUMN_NAME_FROM_INDEX
- GET_COLUMN_VALUE_FROM_INDEX
- GET_COLUMN_VALUE_FROM_NAME
- GET_DATABASE_METADATA
- GET_DDL_RECORD_PROPERTIES
- GETENV
- GET_ENV_VALUE
- GET_ERROR_INFO
- GET_GMT_TIMESTAMP
- GET_MARKER_INFO
- GET_OBJECT_NAME

- GET_OBJECT_NAME_ONLY
- GET_OPERATION_TYPE
- GET_POSITION
- GET_RECORD_BUFFER
- GET_RECORD_LENGTH
- GET_RECORD_TYPE
- GET_SCHEMA_NAME_ONLY
- GET_SESSION_CHARSET
- GET_STATISTICS
- GET_TABLE_COLUMN_COUNT
- GET_TABLE_METADATA
- GET_TABLE_NAME
- GET_TABLE_NAME_ONLY
- GET_TIMESTAMP
- GET_TRANSACTION_IND
- GET_USER_TOKEN_VALUE
- OUTPUT_MESSAGE_TO_REPORT
- RESET_USEREXIT_STATS
- SET_COLUMN_VALUE_BY_INDEX
- SET_COLUMN_VALUE_BY_NAME
- SET_OPERATION_TYPE
- SET_RECORD_BUFFER
- SET_SESSION_CHARSET
- SET_TABLE_NAME

4.3 Using EXIT_CALL_TYPE

Use `EXIT_CALL_TYPE` to indicate when, during processing, the Extract or Replicat process (the caller) calls a user exit routine. A process can call a routine with the following calls.

Table 4-1 User Exit Calls

Call type	Processing point
`EXIT_CALL_ABORT_TRANS`	Valid when the `RECOVERYOPTIONS` mode is `APPEND` (the default). Called when a data pump or Replicat reads a `RESTART ABEND` record from the trail, placed there by a writer process that abended. (The writer process can be the primary Extract writing to a local trail read by a data pump, or a data pump writing to a remote trail read by Replicat.) This call type enables the user exit to abort or discard the transaction that was left incomplete when the writer process stopped, and then to recover and resume processing at the start of the previous completed transaction.

Table 4-1 (Cont.) User Exit Calls

Call type	Processing point
EXIT_CALL_BEGIN_TRANS	Called just before either of the following: • a BEGIN record of a transaction that is read by a data pump • the start of a Replicat transaction
EXIT_CALL_CHECKPOINT	Called just before an Extract or Replicat checkpoint is written.
EXIT_CALL_DISCARD_ASCII_RECORD	Called during Extract processing before an ASCII input record is written to the discard file. The associated ASCII buffer can be retrieved and manipulated by the user exit using callback routines. This call type is not applicable for use with the Replicat process.
EXIT_CALL_DISCARD_RECORD	Called during Replicat processing before a record is written to the discard file. Records can be discarded for several reasons, such as when a value in the Oracle GoldenGate change record is different from the current version in the target table.The associated discard buffer can be retrieved and manipulated by the user exit using callback routines. This call type is not applicable for use with the Extract process.
EXIT_CALL_END_TRANS	Called just after either of the following: • an END record of a transaction that is read by a data pump • the last record in a Replicat transaction
EXIT_CALL_FATAL_ERROR	Called during Extract or Replicat processing just before Oracle GoldenGate terminates after a fatal error.
EXIT_CALL_PROCESS_MARKER	Called during Replicat processing when a marker from a NonStop server is read from the trail, and before writing to the marker history file.
EXIT_CALL_PROCESS_RECORD	• For Extract, called before a record buffer is output to the trail. • For Replicat, called just before a replicated operation is performed. This call is the basis of most user exit processing. When EXIT_CALL_PROCESS_RECORD is called, the record buffer and other record information are available to the user exit through callback routines. If source-target mapping is specified in the parameter file, the mapping is performed before the EXIT_CALL_PROCESS_RECORD event takes place. The user exit can map, transform, clean, or perform virtually any other operation with the record. The user exit can return a status indicating whether the caller should process or ignore the record.
EXIT_CALL_START	Called at the start of processing. The user exit can perform initialization work, such as opening files and initializing variables.
EXIT_CALL_STOP	Called before the process stops gracefully or ends abnormally. The user exit can perform completion work, such as closing files or outputting totals.
EXIT_CALL_RESULT	Set by the user exit routines to instruct the caller how to respond when each exit call completes.

4.51 SET_SESSION_CHARSET

Valid For

Extract and Replicat

Description

Use the SET_SESSION_CHARSET function to set the character set of the user exit. The character set of the user exit session indicates the encoding of any character-based callback structure members that are used between the user exit and the caller process (Extract, data pump, Replicat), including metadata such as (but not limited to):

- database names and locales
- table and column names
- DDL text
- error messages
- character-type columns such as CHAR and NCHAR
- date-time and numeric columns that are represented in string form

This function can be called at any time that the user exit has control. When the user exit sets the session character set, it takes effect immediately, and all character values start being converted to the specified set. The recommended place to call this function is with call type EXIT_CALL_START.

> **Note:**
>
> SET_SESSION_CHARSET is not thread-safe.

If SET_SESSION_CHARSET is not called, the session gets set to the default character set of the operating system, which is a predefined enumerated type value in ULIB_CS_DEFAULT in the ucharset.h file. When the session character set is a default from ULIB_CS_DEFAULT, no conversion is performed by Oracle GoldenGate for character-type values that are exchanged between the user exit and the caller process. In addition, the object-name metadata of the database are considered to be the default character set of the operating system. Keep in mind that the default may not be correct.

The character set of the user exit is printed to the report file when the user exit is loaded and when SET_SESSION_CHARSET is called. If the session character set is ULIB_CS_DEFAULT, there is a message stating that no column data character-set conversion is being performed.

For more information about globalization support, see Administering Oracle GoldenGate for Windows and UNIX.

Syntax

```
#include usrdecs.h
short result_code;
session_def session_charset_def;
ERCALLBACK (SET_SESSION_CHARSET, &session_charset_def, &result_code);
```

Buffer

```
typedef struct
{
```

```
ULibCharSet  session_charset;
} session_def;
```

Input

`session_charset`
The valid values of the session character set are defined in the header file `ucharset.h`.

Output

None

Return Values

`EXIT_FN_RET_OK`

4.1 Calling a User Exit

Write the user exit routine in C programming code. Use the `CUSEREXIT` parameter to call the user exit from a Windows DLL or UNIX shared object at a defined exit point within Oracle GoldenGate processing. Your user exit routine must be able to accept different events and information from the Extract and Replicat processes, process the information as desired, and return a response and information to the caller (the Oracle GoldenGate process that called it). For more information and syntax for the `CUSEREXIT` parameter, see "CUSEREXIT".

4.2 Summary of User Exit Functions

Parameter	Description
`EXIT_CALL_TYPE`	Indicates when, during processing, the routine is called.
`EXIT_CALL_RESULT`	Provides a response to the routine.
`EXIT_PARAMS`	Supplies information to the routine.
`ERCALLBACK`	Implements a callback routine. Callback routines retrieve record and Oracle GoldenGate context information, and they modify the contents of data records.

4.4 Using EXIT_CALL_RESULT

Use `EXIT_CALL_RESULT` to provide a response to the routine.

Table 4-2 User Exit Responses

Call result	Description
`EXIT_ABEND_VAL`	Instructs the caller to terminate immediately.
`EXIT_IGNORE_VAL`	Rejects records for further processing. `EXIT_IGNORE_VAL` is appropriate when the user exit performs all the required processing for a record and there is no need to output or replicate the data record.

Table 4-2 (Cont.) User Exit Responses

Call result	Description
EXIT_OK_VAL	If the routine does nothing to respond to an event, EXIT_OK_VAL is assumed. If the exit call type is any of the following... • EXIT_CALL_PROCESS_RECORD • EXIT_CALL_DISCARD_RECORD • EXIT_CALL_DISCARD_ASCII_RECORD ... and EXIT_OK_VAL is returned, then Oracle GoldenGate processes the record buffer that was returned by the user exit.
EXIT_PROCESSED_REC_VAL	Instructs Extract or Replicat to skip the record, but update the statistics that are printed to the report file for that table and for that operation type.
EXIT_STOP_VAL	Instructs the caller to stop processing gracefully. EXIT_STOP_VAL or EXIT_ABEND_VAL may be appropriate when an error condition occurs in the user exit.

4.5 Using EXIT_PARAMS

Use EXIT_PARAMS to supply information to the user exit routine, such as the program name and user-defined parameters. You can process a single data record multiple times.

Table 4-3 User Exit Input

Exit parameter	Description
PROGRAM_NAME	Specifies the full path and name of the calling process, for example \ggs\extract or \ggs\replicat. Use this parameter when loading an Oracle GoldenGate callback routine using the Windows API or to identify the calling program when user exits are used with both Extract and Replicat processing.
FUNCTION_PARAM	• Allows you to pass a parameter that is a literal string to the user exit. Specify the parameter with the EXITPARAM option of the TABLE or MAP statement from which the parameter will be passed. See "EXITPARAM *'parameter'*". This is only valid during the exit call to process a specific record. • FUNCTION_PARAM can also be used at the exit call startup event to pass the parameters that are specified in the PARAMS option of the CUSEREXIT parameter. (See "CUSEREXIT".) This is only valid to supply a global parameter at exit startup.
MORE_RECS_IND	Set on return from an exit. For database records, determines whether Extract or Replicat processes the record again. This allows the user exit to output many records per record processed by Extract. To request the same record again, set MORE_RECS_IND to CHAR_NO_VAL or CHAR_YES_VAL.

4.6 Using ERCALLBACK

ERCALLBACK is the basic user exit function for Oracle GoldenGate. It is used to pull the record context into user exit. It's like a package that contains multiple individual functions inside it. You can call these functions and get return values. For example,

functions such as GET_BEFORE_AFTER_IND, or GET_COLUMN_VALUE_FROM_NAME can be called. These functions are called function_code.

Syntax

```
ERCALLBACK (function_code, buffer, result_code );
```

function_code
The function to be executed by the callback routine. The user callback routine behaves differently based on the function code passed to the callback routine. While some functions can be used for both Extract and Replicat, the validity of the function in one process or the other is dependent on the input parameters that are set for that function during the callback routine. See Function Codes for a full description of available function codes.

buffer
A void pointer to a buffer containing a predefined structure associated with the specified function code.

result_code
The status of the function executed by the callback routine. The result code returned by the callback routine indicates whether or not the callback function was successful. A result code can be one of the values in Table 4-4.

Table 4-4 Result Codes

Code	Description
EXIT_FN_RET_BAD_COLUMN_DATA	Invalid data was encountered when retrieving or setting column data.
EXIT_FN_RET_BAD_DATE_TIME	A date, timestamp, or interval type of column contains an invalid date or time value.
EXIT_FN_RET_BAD_NUMERIC_VALUE	A numeric type of column contains an invalid numeric value.
EXIT_FN_RET_COLUMN_NOT_FOUND	The column was not found in a compressed update record (update by a database that only logs the values that were changed).
EXIT_FN_RET_ENV_NOT_FOUND	The specified environment value could not be found in the record.
EXIT_FN_RET_EXCEEDED_MAX_LENGTH	The metadata could not be retrieved because the name of the table or column did not fit in the allocated buffer.
EXIT_FN_RET_FETCH_ERROR	The record could not be fetched. View the error message to see the reason.
EXIT_FN_RET_INCOMPLETE_DDL_REC	An internal error occurred when processing the DDL record. The record is probably incomplete.
EXIT_FN_RET_INVALID_CALLBACK_FNC_CD	An invalid callback function code was passed to the callback routine.
EXIT_FN_RET_INVALID_COLUMN	A non-existent column was referred to in the function call.
EXIT_FN_RET_INVALID_COLUMN_TYPE	The routine is trying to manipulate a data type that is not supported by Oracle GoldenGate for that purpose.
EXIT_FN_RET_INVALID_CONTEXT	The callback function was called at an improper time.
EXIT_FN_RET_INVALID_PARAM	An invalid parameter was passed to the callback function.
EXIT_FN_RET_NO_SRCDB_INSTANCE	The source database instance could not be found.
EXIT_FN_RET_NO_TGTDB_INSTANCE	The target database instance could not be found.

Table 4-4 (Cont.) Result Codes

Code	Description
EXIT_FN_RET_NOT_SUPPORTED	This function is not supported for this process.
EXIT_FN_RET_OK	The callback function succeeded.
EXIT_FN_RET_SESSION_CS_CNV_ERR	A ULIB_ERR_INVALID_CHAR_FOUND error was returned to the character-set conversion routine. The conversion failed.
EXIT_FN_RET_TABLE_NOT_FOUND	An invalid table name was specified.
EXIT_FN_RET_TOKEN_NOT_FOUND	The specified user token could not be found in the record.

You can use ERCALLBACK to perform many different function calls. For example, if you want to get the name of a table, you can use the following command:

```
ERCALLBACK (GET_TABLE_NAME, &var, &result_code)
```

These functions are used inside the c code for the user exit to perform any of the calls for functions provided in the section Function Codes. With the combination of the different function_code calls, you can perform many tasks using ERCALLBACK, such as:

- Recreate DML statements
- Perform transformations
- Pull specific columns out of a record
- Write information to a report file

For example, if you need a message written to the report file each time the lag in the heartbeat table exceeds a certain threshold, you could use the CUSEREXIT function. The CUSEREXIT function would then make numerous calls to ERCALLBACK to get the lag column data, perform calculations and the comparison, and if the lag is over the specified threshold then write a message to the report file.

4.7 Function Codes

Function codes determine the output of the callback routine. The callback routine expects the contents of the data buffer to match the structure of the specified function code. The callback routine function codes and their data buffers are described in the following sections. The following is a summary of available functions.

Table 4-5 Summary of Oracle GoldenGate Function Codes

Function code	Description
COMPRESS_RECORD	Use the COMPRESS_RECORD function when some, but not all, of a target table's columns are present after mapping and the entire record must be manipulated, rather than individual column values.
DECOMPRESS_RECORD	Use the DECOMPRESS_RECORD function when some, but not all, of a target table's columns are present after mapping and the entire record must be manipulated, rather than individual column values.

Table 4-5 (Cont.) Summary of Oracle GoldenGate Function Codes

Function code	Description
GET_BASE_OBJECT_NAME	Use the GET_BASE_OBJECT_NAME function to retrieve the fully qualified name of the base object of an object in a record.
GET_BASE_OBJECT_NAME_ONLY	Use the GET_BASE_OBJECT_NAME_ONLY function to retrieve only the name of the base object of an object in a record.
GET_BASE_SCHEMA_NAME_ONLY	Use the GET_BASE_SCHEMA_NAME_ONLY function to retrieve only the name of the schema of the base object of an object in a record.
GET_BEFORE_AFTER_IND	Use the GET_BEFORE_AFTER_IND function to determine whether a record is a before image or an after image of the database operation.
GET_CATALOG_NAME_ONLY	Use the GET_CATALOG_NAME_ONLY function to return the name of the database catalog.
GET_COL_METADATA_FROM_INDEX	Use the GET_COL_METADATA_FROM_INDEX function to determine the column metadata that is associated with a specified column index.
GET_COL_METADATA_FROM_NAME	Use the GET_COL_METADATA_FROM_NAME function to determine the column metadata that is associated with a specified column name.
GET_COLUMN_INDEX_FROM_NAME	Use the GET_COLUMN_INDEX_FROM_NAME function to determine the column index associated with a specified column name.
GET_COLUMN_NAME_FROM_INDEX	Use the GET_COLUMN_NAME_FROM_INDEX function to determine the column name associated with a specified column index.
GET_COLUMN_VALUE_FROM_INDEX	Use the GET_COLUMN_VALUE_FROM_INDEX function to return the column value from the data record using the specified column index.
GET_COLUMN_VALUE_FROM_NAME	Use the GET_COLUMN_VALUE_FROM_NAME function to return the column value from the data record by using the specified column name.
GET_DATABASE_METADATA	Use the GET_DATABASE_METADATA function to return database metadata.
GET_DDL_RECORD_PROPERTIES	Use the GET_DDL_RECORD_PROPERTIES function to return information about a DDL operation.
GET_ENV_VALUE	Use the GET_ENV_VALUE function to return information about the Oracle GoldenGate environment.
GET_ERROR_INFO	Use the GET_ERROR_INFO function to return error information associated with a discard record.
GET_GMT_TIMESTAMP	Use the GET_GMT_TIMESTAMP function to return the operation commit timestamp in GMT format.
GET_MARKER_INFO	Use the GET_MARKER_INFO function to return marker information when posting data. Use markers to trigger custom processing within a user exit.
GET_OBJECT_NAME	Returns the fully qualified two- or three-part name of a table or other object that is associated with the record that is being processed.
GET_OBJECT_NAME_ONLY	Returns the unqualified name of a table or other object that is associated with the record that is being processed.
GET_OPERATION_TYPE	Use the GET_OPERATION_TYPE function to determine the operation type associated with a record.
GET_POSITION	Use the GET_POSITION function is obtain a read position of an Extract data pump or Replicat in the Oracle GoldenGate trail.
GET_RECORD_BUFFER	Use the GET_RECORD_BUFFER function to obtain information for custom column conversions.

Table 4-5 (Cont.) Summary of Oracle GoldenGate Function Codes

Function code	Description
GET_RECORD_LENGTH	Use the GET_RECORD_LENGTH function to return the length of the data record.
GET_RECORD_TYPE	Use the GET_RECORD_TYPE function to return the type of record being processed
GET_SCHEMA_NAME_ONLY	Use the GET_SCHEMA_NAME_ONLY function to return only the schema name of a table.
GET_SESSION_CHARSET	Use the GET_SESSION_CHARSET function to return the character set of the user exit session.
GET_STATISTICS	Use the GET_STATISTICS function to return the current processing statistics for the Extract or Replicat process.
GET_TABLE_COLUMN_COUNT	Use the GET_TABLE_COLUMN_COUNT function to return the total number of columns in a table.
GET_TABLE_METADATA	Use the GET_TABLE_METADATA function to return metadata for the table that associated with the record that is being processed.
GET_TABLE_NAME	Use the GET_TABLE_NAME function to return the fully qualified two- or three-part name of the source or target table that is associated with the record that is being processed.
GET_TABLE_NAME_ONLY	Use the GET_TABLE_NAME_ONLY function to return only the unqualified name of the table that is associated with the record that is being processed.
GET_TIMESTAMP	Use the GET_TIMESTAMP function to return the I/O timestamp associated with a source data record.
GET_TRANSACTION_IND	Use the GET_TRANSACTION_IND function to determine whether a data record is the first, last or middle operation in a transaction,
GET_USER_TOKEN_VALUE	Use the GET_USER_TOKEN_VALUE function to obtain the value of a user token from a trail record.
OUTPUT_MESSAGE_TO_REPORT	Use the OUTPUT_MESSAGE_TO_REPORT function to output a message to the report file.
RESET_USEREXIT_STATS	Use the RESET_USEREXIT_STATS function to reset the statistics for the Oracle GoldenGate process.
SET_COLUMN_VALUE_BY_INDEX	Use the SET_COLUMN_VALUE_BY_INDEX function to modify a single column value without manipulating the entire data record.
STRNCMP	Use the SET_COLUMN_VALUE_BY_NAME function to modify a single column value without manipulating the entire data record.
SET_OPERATION_TYPE	Use the SET_OPERATION_TYPE function to change the operation type associated with a data record.
SET_RECORD_BUFFER	Use the SET_RECORD_BUFFER function for compatibility with HP NonStop user exits, and for complex data record manipulation.
SET_SESSION_CHARSET	Use the SET_SESSION_CHARSET function to set the character set of the user exit session.
SET_TABLE_NAME	Use the SET_TABLE_NAME function to change the table name associated with a data record.

4.8 COMPRESS_RECORD

Valid For

Extract and Replicat

Description

Use the `COMPRESS_RECORD` function to re-compress records that have been decompressed with the `DECOMPRESS_RECORD` function. Call `COMPRESS_RECORD` only *after* using `DECOMPRESS_RECORD`.

The content of the record buffer is not converted to or from the character set of the user exit. It is passed as-is.

Syntax

```
#include "usrdecs.h"
short result_code;
compressed_rec_def compressed_rec;
ERCALLBACK (COMPRESS_RECORD, &compressed_rec, &result_code);
```

Buffer

```
typedef struct
{
char *compressed_rec;
long compressed_len;
char *decompressed_rec;
long decompressed_len;
short *columns_present;
short source_or_target;
char requesting_before_after_ind;
} compressed_rec_def;
```

Input

`decompressed_rec`
A pointer to the buffer containing the record before compression. The record is assumed to be in the default Oracle GoldenGate canonical format.

`decompressed_len`
The length of the decompressed record.

`source_or_target`
One of the following to indicate whether the source or target record is being compressed.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

`requesting_before_after_ind`
Used as internal input. Does not need to be set. If set, it will be ignored.

columns_present
An array of values that indicates 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 in the table is seven, the array should contain:

```
1, 0, 1, 0, 0, 1, 0
```

Use the GET_TABLE_COLUMN_COUNT function to get the number of columns in the table (see "GET_TABLE_COLUMN_COUNT").

Output

compressed_rec
A pointer to the record returned in compressed format. Typically, compressed_rec is a pointer to a buffer of type exit_rec_buf_def. The exit_rec_buf_def buffer contains the actual record about to be processed by Extract or Replicat. The buffer is supplied when the call type is EXIT_CALL_DISCARD_RECORD. Exit routines may change the contents of this buffer, for example to perform custom mapping functions. The caller must ensure that the appropriate amount of memory is allocated to compressed_rec.

compressed_len
The returned length of the compressed record.

Return Values

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_OK
EXIT_FN_RET_INVALID_PARAM
```

4.9 DECOMPRESS_RECORD

Valid For

Extract and Replicat

Description

Use the DECOMPRESS_RECORD function when you want to retrieve or manipulate an entire update record with the GET_RECORD_BUFFER (see "GET_RECORD_BUFFER") or SET_RECORD_BUFFER function (see "SET_RECORD_BUFFER") and the record is compressed. DECOMPRESS_RECORD makes compressed records easier to process and map by putting the record into its logical column layout. The columns that are present will be in the expected positions without the index and length indicators (see "Compressed Record Format"). The missing columns will be represented as zeroes. When used, DECOMPRESS_RECORD should be invoked before any manipulation occurs. After the user exit processing is completed, use the COMPRESS_RECORD function (see "COMPRESS_RECORD") to re-compress the record before returning it to the Oracle GoldenGate process.

This function is valid for processing UPDATE operations only. Deletes, inserts and updates appear in the buffer as full record images.

The content of the record buffer is not converted to or from the character set of the user exit. It is passed as-is.

Compressed Record Format

Compressed SQL updates have the following 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 one of the following two-byte null indicators when applicable. `0` is not null. `-1` is null.

Syntax

```
#include "usrdecs.h"
short result_code;
compressed_rec_def compressed_rec;
ERCALLBACK (DECOMPRESS_RECORD, &compressed_rec, &result_code);
```

Buffer

```
typedef struct
{
char *compressed_rec;
long compressed_len;
char *decompressed_rec;
long decompressed_len;
short *columns_present;
short source_or_target;
char requesting_before_after_ind;
} compressed_rec_def;
```

Input

`compressed_rec`
A pointer to the record in compressed format. Use the `GET_RECORD_BUFFER` function to obtain this value (see "GET_RECORD_BUFFER").

`compressed_len`
The length of the compressed record. Use the `GET_RECORD_BUFFER` (see "GET_RECORD_BUFFER") or `GET_RECORD_LENGTH` (see "GET_RECORD_LENGTH") function to get this value.

`source_or_target`
One of the following to indicate whether the source or target record is being decompressed.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

`requesting_before_after_ind`
Used as internal input. Does not need to be set. If set, it will be ignored.

Output

decompressed_rec
A pointer to the record returned in decompressed format. The record is assumed to be in the Oracle GoldenGate internal canonical format. The caller must ensure that the appropriate amount of memory is allocated to `decompressed_rec`.

decompressed_len
The returned length of the decompressed record.

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 in the table is seven, the array should contain:

```
1, 0, 1, 0, 0, 1, 0
```

This array helps mapping functions determine when and whether a compressed column should be mapped.

Return Values

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_OK
EXIT_FN_RET_INVALID_PARAM
```

4.10 GET_BASE_OBJECT_NAME

Valid For

Extract and Replicat

Description

Use the `GET_BASE_OBJECT_NAME` function to retrieve the fully qualified name of the base object of a source or target object that is associated with the record being processed. This function is valid tables and other objects in a DDL operation.

To return only part of the base object name, see the following:

GET_BASE_OBJECT_NAME_ONLY GET_BASE_SCHEMA_NAME_ONLY

Database object names are returned exactly as they are defined in the hosting database, including the letter case.

Syntax

```
#include "usrdecs.h"
short result_code;
env_value_def env_value;
ERCALLBACK (GET_BASE_OBJECT_NAME, &env_value, &result_code);
```

Buffer

```
typedef struct
{
char *buffer;
long max_length;
```

```
long actual_length;
short value_truncated;
short index;
short source_or_target;
} env_value_def;
```

Input

`buffer`
A pointer to a buffer to accept the returned object name. The name is null-terminated.

`max_length`
The maximum length of your allocated buffer to accept the object name. This is returned as a `NULL` terminated string.

`source_or_target`
One of the following indicating whether to return the source or target object name.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

`buffer`
The fully qualified, null-terminated object name, for example `schema.object` or `catalog.schema.object`, depending on the database platform.
If the character session of the user exit is set with `SET_SESSION_CHARSET` to a value other than the default character set of the operating system, as defined in `ULIB_CS_DEFAULT` in the `ucharset.h` file, the object name is interpreted in the session character set.

`actual length`
The string length of the returned object name. The actual length does not include the null terminator. The actual length is 0 if the object is a table.

`value_truncated`
A flag (`0` or `1`) indicating whether or not the value was truncated. Truncation occurs if the length of the object name plus the null terminator exceeds the maximum buffer length.

Return Values

```
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

4.11 GET_BASE_OBJECT_NAME_ONLY

Valid For

Extract and Replicat

Description

Use the `GET_BASE_OBJECT_NAME_ONLY` function to retrieve the unqualified name (without the catalog, container, or schema) of the base object of a source or target

object that is associated with the record that is being processed. This function is valid for tables and other objects in a DDL operation.

To return the fully qualified name of a base object, see the following:

GET_OBJECT_NAME

To return only the schema of the base object, see the following:

GET_BASE_SCHEMA_NAME_ONLY

Database object names are returned exactly as they are defined in the hosting database, including the letter case.

Syntax

```
#include "usrdecs.h"
short result_code;
env_value_def env_value;
ERCALLBACK (GET_BASE_OBJECT_NAME_ONLY, &env_value, &result_code);
```

Buffer

```
typedef struct
{
char *buffer;
long max_length;
long actual_length;
short value_truncated;
short index;
short source_or_target;
} env_value_def;
```

Input

`buffer`
A pointer to a buffer to accept the returned object name. The name is null-terminated.

`max_length`
The maximum length of your allocated buffer to accept the object name. This is returned as a `NULL` terminated string.

`source_or_target`
One of the following indicating whether to return the source or target object name.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

`buffer`
The fully qualified, null-terminated object name, for example `schema.object` or `catalog.schema.object`, depending on the database platform.
If the character session of the user exit is set with `SET_SESSION_CHARSET` to a value other than the default character set of the operating system, as defined in `ULIB_CS_DEFAULT` in the `ucharset.h` file, the object name is interpreted in the session character set.

actual length
The string length of the returned object name. The actual length does not include the null terminator. The actual length is 0 if the object is a table.

value_truncated
A flag (0 or 1) indicating whether or not the value was truncated. Truncation occurs if the length of the object name plus the null terminator exceeds the maximum buffer length.

Return Values

```
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

4.12 GET_BASE_SCHEMA_NAME_ONLY

Description

Use the GET_BASE_SCHEMA_NAME_ONLY function to retrieve the name of the owner (such as schema), but not the name, of the base object of the source or target object associated with the record being processed. This function is valid for DDL operations.

To return the fully qualified name of a base object, see the following:

GET_BASE_OBJECT_NAME

To return only the unqualified base object name, see the following:

GET_BASE_OBJECT_NAME_ONLY

Database object names are returned exactly as they are defined in the hosting database, including the letter case.

Syntax

```
#include "usrdecs.h"
short result_code;
env_value_def env_value;
ERCALLBACK (GET_BASE_SCHEMA_NAME_ONLY, &env_value, &result_code);
```

Buffer

```
typedef struct
{
char *buffer;
long max_length;
long actual_length;
short value_truncated;
short index;
short source_or_target;
} env_value_def;
```

Input

buffer
A pointer to a buffer to accept the returned schema name. The name is null-terminated.

max_length
The maximum length of your allocated buffer to accept the schema name. This is returned as a NULL terminated string.

source_or_target
One of the following indicating whether to return the source or target schema name.

EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL

Output

buffer
The fully qualified, null-terminated schema name.
If the character session of the user exit is set with SET_SESSION_CHARSET to a value other than the default character set of the operating system, as defined in ULIB_CS_DEFAULT in the ucharset.h file, the schema name is interpreted in the session character set.

actual_length
The string length of the returned name. The actual length does not include the null terminator.

value_truncated
A flag (0 or 1) indicating whether or not the value was truncated. Truncation occurs if the length of the schema name plus the null terminator exceeds the maximum buffer length.

Return Values

EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK

4.13 GET_BEFORE_AFTER_IND

Valid For

Extract and Replicat

Description

Use the GET_BEFORE_AFTER_IND function to determine whether a record is a before image or an after image of the database operation. INSERTs are after images, DELETEs are before images, and UPDATEs can be either before or after images (see the Extract and Replicat parameters GETUPDATEBEFORES and GETUPDATEAFTERS). If the before images of UPDATE operations are being extracted, the before images precede the after images within the same update.

Syntax

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (GET_BEFORE_AFTER_IND, &record, &result_code);
```

Buffer

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

Input

None

Output

`before_after_ind`
One of the following to indicate whether the record is a before or after image.

```
BEFORE_IMAGE_VAL
AFTER_IMAGE_VAL
```

Return Values

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_OK
```

4.14 GET_CATALOG_NAME_ONLY

Valid For

Extract and Replicat

Description

Use the `GET_CATALOG_NAME_ONLY` function to retrieve the name of theOracle CDB container, but not the name of the owner (such as schema) or object, of the source or target object associated with the record being processed. This function is valid for DML and DDL operations.

To return the fully qualified name of a table, see the following:

GET_TABLE_NAME

To return the fully qualified name of a non-table object, such as a user, view or index, see the following:

GET_OBJECT_NAME

To return only the unqualified table or object name, see the following:

GET_TABLE_NAME_ONLY

GET_OBJECT_NAME_ONLY

To return other parts of the table or object name, see the following:

GET_SCHEMA_NAME_ONLY

Database object names are returned exactly as they are defined in the hosting database, including the letter case.

Syntax

```
#include "usrdecs.h"
short result_code;
env_value_def env_value;
ERCALLBACK (GET_CATALOG_NAME_ONLY, &env_value, &result_code);
```

Buffer

```
typedef struct
{
char *buffer;
long max_length;
long actual_length;
short value_truncated;
short index;
short source_or_target;
} env_value_def;
```

Input

`buffer`
A pointer to a buffer to accept the returned catalog name. The name is null-terminated.
If the character session of the user exit is set with `SET_SESSION_CHARSET` to a value other than the default character set of the operating system, as defined in `ULIB_CS_DEFAULT` in the `ucharset.h` file, the catalog name is interpreted in the session character set.

`max_length`
The maximum length of your allocated buffer to accept the name. This is returned as a `NULL` terminated string.

`source_or_target`
One of the following indicating whether to return the source or target table catalog.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

buffer
The fully qualified, null-terminated catalog name.

actual_length
The string length of the returned name. The actual length does not include the null terminator.

value_truncated
A flag (0 or 1) indicating whether or not the value was truncated. Truncation occurs if the length of the catalog name plus the null terminator exceeds the maximum buffer length.

Return Values

```
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

4.15 GET_COL_METADATA_FROM_INDEX

Valid For

Extract and Replicat

Description

Use the GET_COL_METADATA_FROM_INDEX function to retrieve column metadata by specifying the index of the desired column.

Database object names are returned exactly as they are defined in the hosting database, including the letter case.

Syntax

```
#include "usrdecs.h"
short result_code;
col_metadata_def column_meta_rec;
ERCALLBACK (GET_COL_METADATA_FROM_INDEX, &column_meta_rec, &result_code);
```

Buffer

```
typedef struct
{
  short column_index;
  char *column_name;
  long max_name_length;
  short native_data_type;
  short gg_data_type;
  short gg_sub_data_type;
  short is_nullable;
  short is_part_of_key;
  short key_column_index;
  short length;
  short precision;
  short scale;
```

```
  short source_or_target;
} col_metadata_def;
```

Input

column_index
The column index of the column value to be returned.

max_name_length
The maximum length of the returned column name. Typically, the maximum length is the length of the name buffer. Since the returned name is null-terminated, the maximum length should equal the maximum length of the column name.

source_or_target
One of the following to indicate whether the source or target record is being compressed.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

column_name
The column name of the column value to be returned.

actual_name_length
The actual length of the returned name.

value_truncated
A flag (`0` or `1`) to indicate whether or not the value was truncated. Truncation occurs if the length of the column name plus the null terminator exceeds the maximum buffer length.

native_data_type
The native (to the database) data type of the column. Either `native_data_type` or `dd_data_type` is returned, depending on the process, as follows:

- If Extract is making the callback request for a source column, `native_data_type` is returned. If Extract is requesting a mapped target column, `gg_data_type` is returned (assuming there is a target definitions file on the system).
- If an Extract data pump is making the callback request for a source column and there is a local database, `native_data_type` is returned. If there is no database, `gg_data_type` is returned (assuming there is a source definitions file on the system). If the pump is requesting the target column, `gg_data_type` is returned (assuming a target definitions file exists on the system).
- If Replicat is making the callback request for the source column, then `gg_data_type` is returned (assuming a source definitions file exists on the system). If Replicat is requesting the source column and `ASSUMETARGETDEFS` is being used in the parameter file, then `native_data_type` is returned. If Replicat is requesting the target column, `native_data_type` is returned.

gg_data_type
The Oracle GoldenGate data type of the column.

gg_sub_data_type
The Oracle GoldenGate sub-type of the column.

is_nullable
Flag indicating whether the column permits a null value (TRUE or FALSE).

is_part_of_key
Flag (TRUE or FALSE) indicating whether the column is part of the key that is being used by Oracle GoldenGate.

key_column_index
Indicates the order of the columns in the index. For example, the following table has two key columns that exist in a different order from the order in which they are declared in the primary key.

```
CREATE TABLE ABC
(
cust_code        VARCHAR2(4),
name             VARCHAR2(30),
city             VARCHAR2(20),
state            CHAR(2),
PRIMARY KEY (city, cust_code)
USING INDEX
);
```

Executing the callback function for each column in the logical column order returns the following:

- cust_code returns 1
- name returns -1
- city returns 0
- state returns -1

If the column is part of the key, the value returned is the order of the column within the key.
If the column is not part of the key, a value of -1 is returned.

length
Returns the length of the column.

precision
If a numeric data type, returns the precision of the column.

scale
If a numeric data type, returns the scale.

Return Values

```
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_EXCEEDED_MAX_LENGTH
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_OK
```

4.16 GET_COL_METADATA_FROM_NAME

Valid For

Extract and Replicat

Description

Use the GET_COL_METADATA_FROM_NAME function to retrieve column metadata by specifying the name of the desired column. If the character session of the user exit is set with SET_SESSION_CHARSET to a value other than the default character set of the operating system, as defined in ULIB_CS_DEFAULT in the ucharset.h file, the character data that is exchanged between the user exit and the process is interpreted in the session character set.

If the database is case-sensitive, object names must be specified in the same letter case as they are defined in the hosting database; otherwise, the case does not matter.

Syntax

```
#include "usrdecs.h"
short result_code;
col_metadata_def column_meta_rec;
ERCALLBACK (GET_COL_METADATA_FROM_NAME, &column_meta_rec, &result_code);
```

Buffer

```
typedef struct
{
  short column_index;
  char *column_name;
  long max_name_length;
  short native_data_type;
  short gg_data_type;
  short gg_sub_data_type;
  short is_nullable;
  short is_part_of_key;
  short key_column_index;
  short length;
  short precision;
  short scale;
  short source_or_target;
} col_metadata_def;
```

Input

column_name
The column name of the column value to be returned.

max_name_length
The maximum length of the returned column name. Typically, the maximum length is the length of the name buffer. Since the returned name is null-terminated, the maximum length should equal the maximum length of the column name.

source_or_target
One of the following to indicate whether the source or target record is being compressed.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

column_index
The column index of the column value to be returned.

actual_name_length
The actual length of the returned name.

source_or_target
One of the following to indicate whether the source or target record is being compressed.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

value_truncated
A flag (`0` or `1`) to indicate whether or not the value was truncated. Truncation occurs if the length of the column name plus the null terminator exceeds the maximum buffer length.

native_data_type
The native (to the database) data type of the column.

gg_data_type
The Oracle GoldenGate data type of the column.

gg_sub_data_type
The Oracle GoldenGate sub-type of the column.

is_nullable
Flag indicating whether the column permits a null value (`TRUE` or `FALSE`).

is_part_of_key
Flag (`TRUE` or `FALSE`) indicating whether the column is part of the key that is being used by Oracle GoldenGate.

key_column_index
Indicates the order of the columns in the index. For example, the following table has two key columns that are defined in one order in the table and another in the index definition.

```
CREATE TABLE tcustmer
(
cust_code        VARCHAR2(4),
name             VARCHAR2(30),
city             VARCHAR2(20),
state            CHAR(2),
PRIMARY KEY (city, cust_code)
USING INDEX
);
```

The return is as follows:

- `cust_code` returns `1`
- `name` returns `-1`

- `city` returns `0`
- `state` returns `-1`

If the column is part of the key, its order in the index is returned as an integer. If the column is not part of the key, a value of -1 is returned.

`length`
Returns the length of the column.

`precision`
If a numeric data type, returns the precision of the column.

`scale`
If a numeric data type, returns the scale.

Return Values

```
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_EXCEEDED_MAX_LENGTH
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_OK
```

4.17 GET_COLUMN_INDEX_FROM_NAME

Valid For

Extract and Replicat

Description

Use the `GET_COLUMN_INDEX_FROM_NAME` function to determine the column index associated with a specified column name. If the character session of the user exit is set with `SET_SESSION_CHARSET` to a value other than the default character set of the operating system, as defined in `ULIB_CS_DEFAULT` in the `ucharset.h` file, the character data that is exchanged between the user exit and the process is interpreted in the session character set.

If the database is case-sensitive, object names must be specified in the same letter case as they are defined in the hosting database; otherwise, the case does not matter.

Syntax

```
#include "usrdecs.h"
short result_code;
env_value_def env_value;
ERCALLBACK (GET_COLUMN_INDEX_FROM_NAME, &env_value, &result_code);
```

Buffer

```
typedef struct
{
char *buffer;
long max_length;
long actual_length;
short value_truncated;
short index;
```

```
short source_or_target;
} env_value_def;
```

Input

buffer
A pointer to the column name

actual_length
The length of the column name within the buffer.

source_or_target
One of the following to indicate whether to use the source or target table to look up column information.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

index
The returned column index for the specified column name.

Return Values

```
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

4.18 GET_COLUMN_NAME_FROM_INDEX

Valid For

Extract and Replicat

Description

Use the GET_COLUMN_NAME_FROM_INDEX function to determine the column name associated with a specified column index. If the character session of the user exit is set with SET_SESSION_CHARSET to a value other than the default character set of the operating system, as defined in ULIB_CS_DEFAULT in the ucharset.h file, the character data that is exchanged between the user exit and the process is interpreted in the session character set.

Database object names are returned exactly as they are defined in the hosting database, including the letter case.

Syntax

```
#include "usrdecs.h"
short result_code;
env_value_def env_value;
ERCALLBACK (GET_COLUMN_NAME_FROM_INDEX, &env_value, &result_code);
```

Buffer

```
typedef struct
{
char *buffer;
long max_length;
long actual_length;
short value_truncated;
short index;
short source_or_target;
} env_value_def;
```

Input

buffer
A pointer to a buffer to accept the returned column name. The column name is null-terminated.

max_length
The maximum length of your allocated `buffer` to accept the resulting column name. This is returned as a `NULL` terminated string.

index
The column index of the column name to be returned.

source_or_target
One of the following to indicate whether to use the source or target table to look up column information.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

buffer
The null-terminated column name.

actual length
The string length of the returned column name. The actual length does not include the null terminator.

value_truncated
A flag (`0` or `1`) to indicate whether or not the value was truncated. Truncation occurs if the length of the column name plus the null terminator exceeds the maximum buffer length.

Return Values

```
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

4.19 GET_COLUMN_VALUE_FROM_INDEX

Valid For

Extract and Replicat

Description

Use the GET_COLUMN_VALUE_FROM_INDEX function to retrieve the column value from the data record using the specified column index. Column values are the basis for most logic within the user exit. You can base complex logic on the values of individual columns within the data record. You can specify the character format of the returned value.

If the character session of the user exit is set with SET_SESSION_CHARSET to a value other than the default character set of the operating system, as defined in ULIB_CS_DEFAULT in the ucharset.h file, the character data that is exchanged between the user exit and the process is interpreted in the session character set.

A column value is set to the session character set only if the following is true:

- The column value is a SQL character type (CHAR/VARCHAR2/CLOB, NCHAR/NVARCHAR2/NCLOB), a SQL date/timestamp/interval/number type)
- The column_value_mode indicator is set to EXIT_FN_CNVTED_SESS_CHAR_FORMAT.

Syntax

```
#include "usrdecs.h"
short result_code;
column_def column;
ERCALLBACK (GET_COLUMN_VALUE_FROM_INDEX, &column, &result_code);
```

Buffer

```
typedef struct
{
char *column_value;
unsigned short max_value_length;
unsigned short actual_value_length;
short null_value;
short remove_column;
short value_truncated;
short column_index;
char *column_name;
/* Version 3 CALLBACK_STRUCT_VERSION */
short column_value_mode;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION */
char requesting_before_after_ind;
char more_lob_data;
/* Version 3 CALLBACK_STRUCT_VERSION  */
ULibCharSet column_charset;
} column_def;
```

Input

column_value
A pointer to a buffer to accept the returned column value.

max_value_length
The maximum length of the returned column value. Typically, the maximum length is the length of the column value buffer. If ASCII format is specified with `column_value_mode`, the column value is null-terminated and the maximum length should equal the maximum length of the column value.

column_index
The column index of the column value to be returned.

column_value_mode
Indicates the format of the column value.

EXIT_FN_CHAR_FORMAT
ASCII format: The value is a null-terminated ASCII (or EBCDIC) string (with a known exception for the sub-data type `UTF16_BE`, which is converted to UTF8.)

> **Note:**
>
> A column value might be truncated when presented to a user exit, because the value is interpreted as an ASCII string and is supposed to be null-terminated. The first value of `0` becomes the string terminator.

- Dates are in the format `CCYY-MM-DD HH:MI:SS.FFFFFF`, in which the fractional time is database-dependent.
- Numeric values are in their string format. For example, `123.45` is represented as `"123.45"`.
- Non-printable characters or binary values are converted to hexadecimal notation.
- Floating point types are output as null-terminated strings, to the first 14 significant digits.

EXIT_FN_RAW_FORMAT
Internal Oracle GoldenGate canonical format: This format includes a two-byte `NULL` indicator and a two-byte variable data length when applicable. No character-set conversion is performed by Oracle GoldenGate for this format for any character data type.

EXIT_FN_CNVTED_SESS_CHAR_FORMAT
User exit character set: This only applies if the column data type is:

- a character-based type, single or multi-byte
- a numeric type with a string representation

This format is not null-terminated.

source_or_target
One of the following to indicate whether to use the source or the target data record to retrieve the column value.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

requesting_before_after_ind
Set when processing an after image record and you want the before-image column value of either an update or a primary key update.
To get the "before" value of the column while processing an "after image" of a primary key update or a regular (non-key) update record, set the `requesting_before_after_ind` flag to `BEFORE_IMAGE_VAL`.

- To access the before image of the key columns of a primary key update, nothing else is necessary.
- To access non-key columns of a primary key update or any column of a regular update, the before image must be available.

The default setting is `AFTER_IMAGE_VAL` (get the after image of the column) when an explicit input for `requesting_before_after_ind` is not specified.
To make a before image available, you can use the `GETUPDATEBEFORES` parameter or you can use the `INCLUDEUPDATEBEFORES` option within the `CUSEREXIT` parameter statement.
Note that:

- `GETUPDATEBEFORES` causes an Extract process to write before-image records to the trail and also to make an `EXIT_CALL_PROCESS_RECORD` call to the user exit with the before images.
- `INCLUDEUPDATEBEFORES` does not cause an `EXIT_CALL_PROCESS_RECORD` call to the user exit nor, in the case of Extract, does it cause the process to write the before image to the trail.

requesting_before_after_ind
To get the before image of the column, set the `char requesting_before_after_ind` flag to `BEFORE_IMAGE_VAL`. To get the after image, set it to `AFTER_IMAGE_VAL`. The default is to always work with the after image unless the before is specified.
To make the before images available, you can use the `GETUPDATEBEFORES` parameter for the `TABLE` statement that contains the table, or you can use the `INCLUDEUPDATEBEFORES` option within the `CUSEREXIT` parameter statement. Both will cause the same callout to the user exit for `process_record`.

Output

column_value
A pointer to the returned column value. If `column_value_mode` is specified as `EXIT_FN_CHAR_FORMAT`, the column value is returned as a null-terminated ASCII string; otherwise, the column value is returned in the Oracle GoldenGate internal canonical format. In ASCII format, dates are returned in the following format:

```
YYYY-MM-DD HH:MI:SS.FFFFFF
```

The inclusion of fractional time is database-dependent.

actual_value_length
The string length of the returned column name, in bytes. The actual length does not include a null terminator when column_value_mode is specified as EXIT_FN_CHAR_FORMAT.

null_value
A flag (0 or 1) indicating whether or not the column value is null. If the null_value flag is 1, then the column value buffer is filled with null bytes.

value_truncated
A flag (0 or 1) indicating whether or not the value was truncated. Truncation occurs if the length of the column value exceeds the maximum buffer length. If column_value_mode was specified as EXIT_FN_CHAR_FORMAT, the null terminator is included in the length of the column.

char more_lob_data
A flag that indicates if more LOB data is present beyond the initial 4K that can be stored in the base record. When a LOB is larger than the 4K limit, it is stored in LOB fragments.
You must allocate the appropriate amount of memory to contain the returned values. Oracle GoldenGate will access LOB columns up to 8K of data at all times, filling up the buffer to the amount that the user exit has allocated. If the LOB is larger than that which was allocated, subsequent callbacks are required to obtain the total column data, until all data has been sent to the user exit.
To determine the end of the data, evaluate more_lob_data. The user exit sets this flag to either CHAR_NO_VAL or CHAR_YES_VAL before accessing a new column. If this flag is still initialized after first callback and is not set to either CHAR_YES_VAL or CAR_NO_VAL, then one of the following is true:

- Enough memory was allocated to handle the LOB.
- It is not a LOB.
- It was not over the 4K limit of the base trail record size.

It is recommended that you obtain the source table metadata to determine if a column might be a LOB.

Return Values

```
EXIT_FN_RET_BAD_COLUMN_DATA
EXIT_FN_RET_COLUMN_NOT_FOUND
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

4.20 GET_COLUMN_VALUE_FROM_NAME

Valid For

Extract and Replicat

Description

Use the GET_COLUMN_VALUE_FROM_NAME function to retrieve the column value from the data record by using the specified column name. Column values are the basis for

most logic within the user exit. You can base complex logic on the values of individual columns within the data record.

If the character session of the user exit is set with SET_SESSION_CHARSET to a value other than the default character set of the operating system, as defined in ULIB_CS_DEFAULT in the ucharset.h file, the character data that is exchanged between the user exit and the process is interpreted in the session character set.

A column value is set to the session character set only if the following is true:

- The column value is a SQL character type (CHAR/VARCHAR2/CLOB, NCHAR/NVARCHAR2/NCLOB), a SQL date/timestamp/interval/number type)
- The column_value_mode indicator is set to EXIT_FN_CNVTED_SESS_CHAR_FORMAT.

If the database is case-sensitive, object names must be specified in the same letter case as they are defined in the hosting database; otherwise, the case does not matter.

Syntax

```
#include "usrdecs.h"
short result_code;
column_def column;
ERCALLBACK (GET_COLUMN_VALUE_FROM_NAME, &column, &result_code);
```

Buffer

```
typedef struct
{
char *column_value;
unsigned short max_value_length;
unsigned short actual_value_length;
short null_value;
short remove_column;
short value_truncated;
short column_index;
char *column_name;
/* Version 3 CALLBACK_STRUCT_VERSION */
short column_value_mode;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION */
char requesting_before_after_ind;
char more_lob_data;
/* Version 3 CALLBACK_STRUCT_VERSION  */
ULibCharSet column_charset;
} column_def;
```

Input

column_value
A pointer to a buffer to accept the returned column value.

max_value_length
The maximum length of the returned column value. Typically, the maximum length is the length of the column value buffer. If ASCII format is specified (see column_value_mode) the column value is null-terminated, and the maximum length should equal the maximum length of the column value.

column_name
The name of the column for the column value to be returned.

column_value_mode
Indicates the character set of the column value.

EXIT_FN_CHAR_FORMAT
ASCII format: The value is a null-terminated ASCII (or EBCDIC) string (with a known exception for the sub-data type `UTF16_BE`, which is converted to UTF8.)

> **Note:**
>
> A column value might be truncated when presented to a user exit, because the value is interpreted as an ASCII string and is supposed to be null-terminated. The first value of `0` becomes the string terminator.

- Dates are in the format `CCYY-MM-DD HH:MI:SS.FFFFFF`, in which the fractional time is database-dependent.
- Numeric values are in their string format. For example, `123.45` is represented as `"123.45"`.
- Non-printable characters or binary values are converted to hexadecimal notation.
- Floating point types are output as null-terminated strings, to the first 14 significant digits.

EXIT_FN_RAW_FORMAT
Internal Oracle GoldenGate canonical format: This format includes a two-byte null indicator and a two-byte variable data length when applicable. No character-set conversion is performed by Oracle GoldenGate for this format for any character data type.

EXIT_FN_CNVTED_SESS_CHAR_FORMAT
User exit character set: This only applies if the column data type is:

- a character-based type, single or multi-byte
- a numeric type with a string representation

This format is not null-terminated.

source_or_target
One of the following indicating whether to use the source or target data record to retrieve the column value.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

requesting_before_after_ind
Set when processing an after image record and you want the before columns of either an update or a primary key update.
To get the "before" value of the column while processing an "after image" of a primary key update or a regular (non-key) update record, set the `requesting_before_after_ind` flag to `BEFORE_IMAGE_VAL`.

- To access the before image of the key columns of a primary key update, nothing else is necessary.
- To access non-key columns of a primary key update or any column of a regular update, the before image must be available.

The default setting is `AFTER_IMAGE_VAL` (get the after image of the column) when an explicit input for `requesting_before_after_ind` is not specified.
To make a before image available, you can use the `GETUPDATEBEFORES` parameter or you can use the `INCLUDEUPDATEBEFORES` option within the `CUSEREXIT` parameter statement.
Note that:

- `GETUPDATEBEFORES` causes an Extract process to write before-image records to the trail and also to make an `EXIT_CALL_PROCESS_RECORD` call to the user exit with the before images.
- `INCLUDEUPDATEBEFORES` does not cause an `EXIT_CALL_PROCESS_RECORD` call to the user exit nor, in the case of Extract, does it cause the process to write the before image to the trail.

Output

`column_value`
A pointer to the returned column value. If `column_value_mode` is specified as `EXIT_FN_CHAR_FORMAT`, the column value is returned as a null-terminated ASCII string; otherwise, the column value is returned in the Oracle GoldenGate internal canonical format. In ASCII format, dates are returned in the following format:

```
CCYY-MM-DD HH:MI:SS.FFFFFF
```

The inclusion of fractional time is database-dependent.

`actual length`
The string length of the returned column name. The actual length does not include a null terminator when `column_value_mode` is specified as `EXIT_FN_CHAR_FORMAT`.

`null_value`
A flag (`0` or `1`) indicating whether or not the column value is null. If the `null_value` flag is `1`, then the column value buffer is filled with null bytes.

`value_truncated`
A flag (`0` or `1`) indicating whether or not the value was truncated. Truncation occurs if the length of the column value exceeds the maximum buffer length. If `column_value_mode` was specified as `EXIT_FN_CHAR_FORMAT`, the null terminator is included in the length of the column.

`char more_lob_data`
A flag that indicates if more LOB data is present beyond the initial 4K that can be stored in the base record. When a LOB is larger than the 4K limit, it is stored in LOB fragments.
You must allocate the appropriate amount of memory to contain the returned values. Oracle GoldenGate will access LOB columns up to 8K of data at all times, filling up the buffer to the amount that the user exit has allocated. If the LOB is larger than that which was allocated, subsequent callbacks are required to obtain the total column data, until all data has been sent to the user exit.

To determine the end of the data, evaluate `more_lob_data`. The user exit sets this flag to either `CAR_NO_VAL` or `CHAR_YES_VAL` before accessing a new column. If this flag is still initialized after first callback and is not set to either `CHAR_YES_VAL` or `CAR_NO_VAL`, then one of the following is true:

- Enough memory was allocated to handle the LOB.
- It is not a LOB.
- It was not over the 4K limit of the base trail record size.

It is recommended that you obtain the source table metadata to determine if a column might be a LOB.

Return Values

```
EXIT_FN_RET_BAD_COLUMN_DATA
EXIT_FN_RET_COLUMN_NOT_FOUND
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

Example

```
memset (&col_meta, 0, sizeof(col_meta));
if (record.mapped)
col_meta.source_or_target = EXIT_FN_TARGET_VAL;
else
col_meta.source_or_target = EXIT_FN_SOURCE_VAL;
col_meta.source_or_target = EXIT_FN_SOURCE_VAL;
col_meta.column_name = (char *)malloc(100);
col_meta.max_name_length = 100;
col_meta.column_index = 1;

call_callback (GET_COL_METADATA_FROM_NAME, &col_meta, &result_code);
```

4.21 GET_DATABASE_METADATA

Valid For

Extract and Replicat

Description

Use the `GET_DATABASE_METADATA` function to return the metadata of the database that is associated with a record.

Buffer

```
typedef struct
{
char*    dbName;
long      dbName_max_length;
long      dbName_actual_length;
unsigned char    dbNameMetadata[MAXDBOBJTYPE];
char*   locale;
long     locale_max_length;
long     locale_actual_length;
} database_def;
```

```
typedef struct
{
    database_def  source_db_def;
    database_def  target_db_def;
} database_defs;
```

Input

dbname
A pointer to a buffer to accept the database name.

dbname_max_length
The maximum length of the buffer to hold the name.

dbname_actual_length
The actual length of the database name.

dbNameMetadata
The name metadata for case-sensitivity, which is the same value that is written by Extract and the data pump to a trail. See *Administering Oracle GoldenGate* for a list of macros that can be used by the user exit to check database object name metadata, given an object name type.

locale
A null-terminated character string specifying the locale of the database. This is returned as a conjunction of:

- ISO-639 two-letter language code
- ISO-3166 two-letter country code
- Variant code using '_' U+005F as separator.

Example: `"en_US", "ja_Japen"`

locale_max_length
The maximum length of the buffer to accept the locale.

locale_actual_length
The actual length of the locale.

database_def source_db_def
Directs the process to return metadata for the source database.

database_def target_db_def
Directs the process to return metadata for the target database.

4.22 GET_DDL_RECORD_PROPERTIES

Valid For

Extract and Replicat, for databases for which DDL replication is supported

Description

Use the `GET_DDL_RECORD_PROPERTIES` function to return a DDL operation, including information about the object on which the DDL was performed and also the text of the DDL statement itself. The Extract process can only get the source table layout. The Replicat process can get source or target layouts.

If the character session of the user exit is set with SET_SESSION_CHARSET to a value other than the default character set of the operating system, as defined in ULIB_CS_DEFAULT in the ucharset.h file, the character data that is exchanged between the user exit and the process is interpreted in the session character set. This includes the DDL type, the object type, the two- or three-part object name, the owner name and the DDL text itself.

```
#include "usrdecs.h"
short result_code;
ddl_record_def ddl_rec;
ERCALLBACK (GET_DDL_RECORD_PROPERTIES, &ddl_rec, &result_code);
```

Buffer

```
typedef struct
{
char *ddl_type;
long ddl_type_max_length; /* Maximum Description length PASSED IN BY USER */
long ddl_type_length; /* Actual length */

char *object_type;
long object_type_max_length; /* Maximum Description length PASSED IN BY USER */
long object_type_length; /* Actual length */

char *object_name; /* Fully qualified name of the object
  (3-part for CDB, 2-part for non-CDB) */
long object_max_length; /* Maximum Description length PASSED IN BY USER */
long object_length; /* Actual length */

char *owner_name;
long owner_max_length; /* Maximum Description length PASSED IN BY USER */
long owner_length; /* Actual length */

char *ddl_text;
long ddl_text_max_length; /* Maximum Description length PASSED IN BY USER */
long ddl_text_length; /* Actual length */

short ddl_text_truncated; /* Was value truncated? */
short source_or_target; /* Source or target value? */
} ddl_record_def;
```

Input

ddl_type_length
object_type_length
object_length
owner_length
ddl_text_length
A pointer to one buffer for each of these items to accept the returned column values. These items are as follows:

ddl_type_length
Contains the length of the type of DDL operation, for example a CREATE or ALTER.

object_type_length
Contains the length of type of database object that is affected by the DDL operation, for example TABLE or INDEX.

object_length
Contains the length of the name of the object.

object_length
Contains the length of the owner of the object (schema or database).

ddl_text_length
Contains the length of the actual DDL statement text.

ddl_type_max_length
The maximum length of the DDL operation type that is returned by `*ddl_type`. The DDL type is any DDL command that is valid for the database, such as `ALTER`.

object_type_max_length
The maximum length of the object type that is returned by `*object_type`. The object type is any object that is valid for the database, such as `TABLE`, `INDEX`, and `TRIGGER`.

object_max_length
The maximum length of the name of the object that is returned by `*object_name`.

owner_max_length
The maximum length of the name of the owner that is returned by `*owner_name`.

ddl_text_max_length
The maximum length of the text of the DDL statement that is returned by `*ddl_text`.

source_or_target
One of the following indicating whether to return the operation type for the source or the target data record.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

ddl_type_length
object_type_length
object_length
owner_length
ddl_text_length
All of these fields return the actual length of the value that was requested. (See the input for descriptions.)

ddl_text_truncated
A flag (`0` or `1`) to indicate whether or not the DDL text was truncated. Truncation occurs if the length of the DDL text plus the null terminator exceeds the maximum buffer length.

Return Values

```
EXIT_FN_RET_OK
EXIT_FN_RET_NOT_SUPPORTED
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INCOMPLETE_DDL_REC
```

4.23 GETENV

Use the @GETENV function to return information about the Oracle GoldenGate environment. You can use the information as input into the following:

- Stored procedures or queries (with SQLEXEC)
- Column maps (with the COLMAP option of TABLE or MAP)
- User tokens (defined with the TOKENS option of TABLE and mapped to target columns by means of the @TOKEN function)
- The GET_ENV_VALUE user exit function (see "GET_ENV_VALUE")

> **Note:**
>
> All syntax options must be enclosed within quotes as shown in the syntax descriptions.

Syntax

```
@GETENV (
'LAG' , 'unit' |
'LASTERR' , 'error_info' |
'JULIANTIMESTAMP' |
'JULIANTIMESTAMP_PRECISE' |
'RECSOUTPUT' |
{'STATS'|'DELTASTATS'}, ['TABLE', 'table'], 'statistic' |
'GGENVIRONMENT', 'environment_info' |
'GGFILEHEADER', 'header_info' |
'GGHEADER', 'header_info' |
'RECORD', 'location_info' |
'DBENVIRONMENT', 'database_info'
'TRANSACTION', 'transaction_info' |
'OSVARIABLE', 'variable' |
'TLFKEY', SYSKEY, unique_key
'USERNAME',
'OSUSERNAME',
'MACHINENAME',
'PROGRAMNAME',
'CLIENTIDENTIFIER',
)
```

'LAG' , 'unit'

Valid for Extract and Replicat.

Use the LAG option of @GETENV to return lag information. Lag is the difference between the time that a record was processed by Extract or Replicat and the timestamp of that record in the data source.

Syntax

```
@GETENV ('LAG', {'SEC'|'MSEC'|'MIN'})
```

'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.

'LASTERR' , '*error_info*'

Valid for Replicat.

Use the LASTERR option of @GETENV to return information about the last failed operation processed by Replicat.

Syntax

```
@GETENV ('LASTERR', {'DBERRNUM'|'DBERRMSG'|'OPTYPE'|'ERRTYPE'})
```

'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. For a list of Oracle GoldenGate operation types, see *Administering Oracle GoldenGate*.

'ERRTYPE'
Returns the type of error. Possible results are:

- DB (for database errors)
- MAP (for errors in mapping)

'JULIANTIMESTAMP' | 'JULIANTIMESTAMP_PRECISE'

Valid for Extract and Replicat.

Use the JULIANTIMESTAMP option of @GETENV to return the current time in Julian format. The unit is microseconds (one millionth of a second). On a Windows machine, the value is padded with zeros (0) because the granularity of the Windows timestamp is milliseconds (one thousandth of a second). For example, the following is a typical column mapping:

```
MAP dbo.tab8451, Target targ.tabjts, COLMAP (USEDEFAULTS, &
JTSS = @GETENV ('JULIANTIMESTAMP')
JTSFFFFFF = @date ('yyyy-mm-dd hh:mi:ss.ffffff', 'JTS', &
@getenv ('JULIANTIMESTAMP') ) )
;
```

Possible values that the JTSS and JTSFFFFFF columns can have are:

```
212096320960773000 2010-12-17:16:42:40.773000
212096321536540000 2010-12-17:16:52:16.540000
212096322856385000 2010-12-17:17:14:16.385000
```

```
212096323062919000 2010-12-17:17:17:42.919000
212096380852787000 2010-12-18:09:20:52.787000
```

The last three digits (the microseconds) of the number all contain the padding of 0s .

Optionally, you can use the `'JULIANTIMESTAMP_PRECISE'` option to obtain a timestamp with high precision though this may effect performance.

> **Note:**
>
> Do not use these values for ordering operations.
> Instead use this value: `@COMPUTE(@COMPUTE(@NUMSTR(@GETENV ("RECORD", "FILESEQNO")*100000000000)+@NUMSTR(@GETENV ("RECORD", "FILERBA")))"`

Syntax

```
@GETENV ('JULIANTIMESTAMP')
@GETENV ('JULIANTIMESTAMP_PRECISE')
```

`'RECSOUTPUT'`

Valid for Extract.

Use the `RECSOUTPUT` option of `@GETENV` to retrieve a current count of the number of records that Extract has written to the trail file since the process started. The returned value is not unique to a table or transaction, but instead for the Extract session itself. The count resets to 1 whenever Extract stops and then is started again.

Syntax

```
@GETENV ('RECSOUTPUT')
```

`{'STATS'|'DELTASTATS'}, ['TABLE', '`*`table`*`'], '`*`statistic`*`'`

Valid for Extract and Replicat.

Use the `STATS` and `DELTASTATS` options of `@GETENV` to return the number of operations that were processed per table for any or all of the following:

- `INSERT` operations
- `UPDATE` operations
- `DELETE` operations
- `TRUNCATE` operations
- Total DML operations
- Total DDL operations
- Number of conflicts that occurred, if the Conflict Detection and Resolution (CDR) feature is used.
- Number of CDR resolutions that succeeded
- Number of CDR resolutions that failed

Any errors in the processing of this function, such as an unresolved table entry or incorrect syntax, returns a zero (0) for the requested statistics value.

Understanding How Recurring Table Specifications Affect Operation Counts

An Extract that is processing the same source table to multiple output trails returns statistics based on each localized output trail to which the table linked to @GETENV is written. For example, if Extract captures 100 inserts for table ABC and writes table ABC to three trails, the result for the @GETENV is 300

```
EXTRACT ABC
...
EXTTRAIL c:\ogg\dirdat\aa;
TABLE TEST.ABC;
EXTTRAIL c:\ogg\dirdat\bb;
TABLE TEST.ABC;
TABLE EMI, TOKENS (TOKEN-CNT = @GETENV ('STATS', 'TABLE', 'ABC', 'DML'));
EXTTRAIL c:\ogg\dirdat\cc;
TABLE TEST.ABC;
```

In the case of an Extract that writes a source table multiple times to a single output trail, or in the case of a Replicat that has multiple MAP statements for the same TARGET table, the statistics results are based on all matching TARGET entries. For example, if Replicat filters 20 rows for REGION 'WEST,' 10 rows for REGION 'EAST,' 5 rows for REGION 'NORTH,' and 2 rows for REGION 'SOUTH' (all for table ABC) the result of the @GETENV is 37.

```
REPLICAT ABC
...
MAP TEST.ABC, TARGET TEST.ABC, FILTER (@STREQ (REGION, 'WEST'));
MAP TEST.ABC, TARGET TEST.ABC, FILTER (@STREQ (REGION, 'EAST'));
MAP TEST.ABC, TARGET TEST.ABC, FILTER (@STREQ (REGION, 'NORTH'));
MAP TEST.ABC, TARGET TEST.ABC, FILTER (@STREQ (REGION, 'SOUTH'));
MAP TEST.EMI, TARGET TEST.EMI, &
    COLMAP (CNT = @GETENV ('STATS', 'TABLE', 'ABC', 'DML'));
```

Capturing Multiple Statistics

You can execute multiple instances of @GETENV to get counts for different operation types.

This example returns statistics only for INSERT and UPDATE operations:

```
REPLICAT TEST
..
..
MAP TEST.ABC, TARGET TEST.ABC, COLMAP (USEDEFAULTS, IU = @COMPUTE (@GETENV &
    ('STATS', 'TABLE', 'ABC', 'DML') - (@GETENV ('STATS', 'TABLE', &
    'ABC', 'DELETE'));
```

This example returns statistics for DDL and TRUNCATE operations:

```
REPLICAT TEST2
..
..
MAP TEST.ABC, TARGET TEST.ABC, COLMAP (USEDEFAULTS, DDL = @COMPUTE &
(@GETENV ('STATS', 'DDL') + (@GETENV ('STATS', 'TRUNCATE'));
```

Example Use Case

In the following use case, if all DML from the source is applied successfully to the target, Replicat suspends by means of EVENTACTIONS with SUSPEND, until resumed from GGSCI with SEND REPLICAT with RESUME.

GETENV used in Extract parameter file:

```
TABLE HR1.HR*;
TABLE HR1.STAT, TOKENS ('env_stats' = @GETENV ('STATS', 'TABLE', &
    'HR1.HR*', 'DML'));
```

GETENV used in Replicat parameter file:

```
MAP HR1.HR*, TARGET HR2.*;
MAP HR1.STAT, TARGET HR2.STAT, filter (
    @if (
    @token ('stats') =
    @getenv ('STATS', 'TABLE', 'TSSCAT.TCUSTORD', 'DML'), 1, 0 )
    ),
    eventactions (suspend);
```

Using Statistics in FILTER Clauses

Statistics returned by STATS and DELTASTATS are dynamic values and are incremented after mapping is performed. Therefore, when using CDR statistics in a FILTER clause in each of multiple MAP statements, you need to order the MAP statements in descending order of the statistics values. If the order is not correct, Oracle GoldenGate returns error OGG-01921. For detailed information about this requirement, see Document 1556241.1 in the Knowledge base of My Oracle Support at http://support.oracle.com.

Example 4-1 MAP statements containing statistics in FILTER clauses

In the following example, the MAP statements containing the filter for the CDR_CONFLICTS statistic are ordered in descending order of the statistic: >3, then =3, then <3.

```
MAP TEST.GG_HEARTBEAT_TABLE, TARGET TEST.GG_HEARTBEAT_TABLE COMPARECOLS
(ON UPDATE ALL),RESOLVECONFLICT(UPDATEROWEXISTS,(DEFAULT, OVERWRITE)),FILTER
(@GETENV ("STATS", "CDR_CONFLICTS") > 3),EVENTACTIONS (LOG INFO);MAP
TEST.GG_HEARTBEAT_TABLE, TARGET TEST.GG_HEARTBEAT_TABLE COMPARECOLS (ON
UPDATE ALL),RESOLVECONFLICT(UPDATEROWEXISTS,(DEFAULT, OVERWRITE)),FILTER
(@GETENV ("STATS", "CDR_CONFLICTS") = 3),EVENTACTIONS (LOG WARNING);MAP
TEST.GG_HEARTBEAT_TABLE, TARGET TEST.GG_HEARTBEAT_TABLE COMPARECOLS (ON UPDATE
ALL),RESOLVECONFLICT(UPDATEROWEXISTS,(DEFAULT, OVERWRITE)),FILTER (@GETENV
("STATS", "CDR_CONFLICTS") < 3),EVENTACTIONS (LOG WARNING);
```

Syntax

```
@GETENV ({'STATS' | 'DELTASTATS'}, ['TABLE', 'table'], 'statistic')
```

{'STATS' | 'DELTASTATS'}
STATS returns counts since process startup, whereas DELTASTATS returns counts since the last execution of a DELTASTATS.
The execution logic is as follows:

- When Extract processes a transaction record that satisfies `@GETENV` with `STATS` or `DELTASTATS`, the table name is matched against resolved source tables in the `TABLE` statement.
- When Replicat processes a trail record that satisfies `@GETENV` with `STATS` or `DELTASTATS`, the table name is matched against resolved target tables in the `TARGET` clause of the `MAP` statement.

`'TABLE', 'table'`
Executes the `STATS` or `DELTASTATS` only for the specified table or tables. Without this option, counts are returned for all tables that are specified in `TABLE` (Extract) or `MAP` (Replicat) parameters in the parameter file.
Valid *`table_name`* values are:

- '*`schema.table`*' specifies a table.
- *`'table'`* specifies a table of the default schema.
- '*`schema.*`*' specifies all tables of a schema.
- '*' specifies all tables of the default schema.

For example, the following counts DML operations only for tables in the `hr` schema:

```
MAP fin.*, TARGET fin.*;
MAP hr.*, TARGET hr.*;
MAP hq.rpt, TARGET hq.rpt, COLMAP (USEDEFAULTS, CNT = @GETENV ('STATS',
'TABLE', 'hr.*', 'DML'));
```

Likewise, the following counts DML operations only for the `emp` table in the `hr` schema:

```
MAP fin.*, TARGET fin.*;
MAP hr.*, TARGET hr.*;
MAP hq.rpt, TARGET hq.rpt, COLMAP (USEDEFAULTS, CNT = @GETENV ('STATS',
'TABLE', 'hr.emp', 'DML'));
```

By contrast, because there are no specific tables specified for `STATS` in the following example, the function counts all `INSERT`, `UPDATE`, and `DELETE` operations for all tables in all schemas that are represented in the `TARGET` clauses of `MAP` statements:

```
MAP fin.*, TARGET fin.*;
MAP hr.*, TARGET hr.*;
MAP hq.rpt, TARGET hq.rpt, COLMAP (USEDEFAULTS, CNT = &
@GETENV ('STATS', 'DML'));
```

`'statistic'`
The type of statistic to return. See Using Statistics in FILTER Clauses for important information when using statistics in `FILTER` clauses in multiple `TABLE` or `MAP` statements.

`'INSERT'`
Returns the number of `INSERT` operations that were processed.

`'UPDATE'`
Returns the number of `UPDATE` operations that were processed.

`'DELETE'`
Returns the number of `DELETE` operations that were processed.

'DML'
Returns the total of INSERT, UPDATE, and DELETE operations that were processed.

'TRUNCATE'
Returns the number of TRUNCATE operations that were processed. This variable returns a count only if Oracle GoldenGate DDL replication is not being used. If DDL replication is being used, this variable returns a zero.

'DDL'
Returns the number of DDL operations that were processed, including TRUNCATEs and DDL specified in INCLUDE and EXCLUDE clauses of the DDL parameter, all scopes (MAPPED, UNMAPPED, OTHER). This variable returns a count only if Oracle GoldenGate DDL replication is being used. This variable is not valid for 'DELTASTATS'.

'CDR_CONFLICTS'
Returns the number of conflicts that Replicat detected when executing the Conflict Detection and Resolution (CDR) feature.
Example for a specific table:

```
@GETENV ('STATS','TABLE','HR.EMP','CDR_CONFLICTS')
```

Example for all tables processed by Replicat:

```
@GETENV ('STATS','CDR_CONFLICTS')
```

'CDR_RESOLUTIONS_SUCCEEDED'
Returns the number of conflicts that Replicat resolved when executing the Conflict Detection and Resolution (CDR) feature.
Example for a specific table:

```
@GETENV ('STATS','TABLE','HR.EMP', 'CDR_RESOLUTIONS_SUCCEEDED')
```

Example for all tables processed by Replicat:

```
@GETENV ('STATS','CDR_RESOLUTIONS_SUCCEEDED')
```

'CDR_RESOLUTIONS_FAILED'
Returns the number of conflicts that Replicat could not resolve when executing the Conflict Detection and Resolution (CDR) feature.
Example for a specific table:

```
@GETENV ('STATS','TABLE','HR.EMP', 'CDR_RESOLUTIONS_FAILED')
```

Example for all tables processed by Replicat:

```
@GETENV ('STATS','CDR_RESOLUTIONS_FAILED')
```

'GGENVIRONMENT' , '*environment_info*'

Valid for Extract and Replicat.

Use the GGENVIRONMENT option of @GETENV to return information about the Oracle GoldenGate environment.

Syntax

```
@GETENV ('GGENVIRONMENT', {'DOMAINNAME'|'GROUPDESCRIPTION'|'GROUPNAME'|
   'GROUPTYPE'|'HOSTNAME'|'OSUSERNAME'|'PROCESSID')
```

'DOMAINNAME'
(Windows only) Returns the domain name associated with the user that started the process.

'GROUPDESCRIPTION'
Returns the description of the group, taken from the checkpoint file. Requires that a description was provided with the `DESCRIPTION` parameter when the group was created with the `ADD` command in GGSCI.

'GROUPNAME'
Returns the name of the process group.

'GROUPTYPE'
Returns the type of process, either `EXTRACT` or `REPLICAT`.

'HOSTNAME'
Returns the name of the system running the Extract or Replicat process.

'OSUSERNAME'
Returns the operating system user name that started the process.

'PROCESSID'
Returns the process ID that is assigned to the process by the operating system.

'GGHEADER' , '*header_info*'

Valid for Extract and Replicat.

Use the `GGHEADER` option of `@GETENV` to return information from the header portion of an Oracle GoldenGate trail record. The header describes the transaction environment of the record. For more information on record headers and record types, see *Administering Oracle GoldenGate*.

Syntax

```
@GETENV ('GGHEADER', {'BEFOREAFTERINDICATOR'|'COMMITTIMESTAMP'|'LOGPOSITION'|
   'LOGRBA'|'OBJECTNAME'|'TABLENAME'|'OPTYPE'|'RECORDLENGTH'|
   'TRANSACTIONINDICATOR'})
```

> **Note:**
>
> Do not use `TIMESTAMP_PRECISE` for ordering operations.
> Instead use this value: `@COMPUTE(@COMPUTE(@NUMSTR(@GETENV ("RECORD", "FILESEQNO"))*100000000000)+@NUMSTR(@GETENV ("RECORD", "FILERBA")))`

'BEFOREAFTERINDICATOR'
Returns the before or after indicator showing whether the record is a before image or an after image. Possible results are:

- `BEFORE` (before image)
- `AFTER` (after image)

'COMMITTIMESTAMP'
Returns the transaction timestamp (the time when the transaction committed) expressed in the format of YYYY-MM-DD HH:MI:SS.FFFFFF, for example:

```
2011-01-24 17:08:59.000000
```

'LOGPOSITION'
Returns the position of the Extract process in the data source. (See the LOGRBA option.)

'LOGRBA'
LOGRBA and LOGPOSITION store details of the position in the data source of the record. For transactional log-based products, LOGRBA is the sequence number and LOGPOSITION is the relative byte address. However, these values will vary depending on the capture method and database type.

'OBJECTNAME' | 'TABLENAME'
Returns the table name or object name (if a non-table object).

'OPTYPE'
Returns the type of operation. Possible results are:

```
INSERT
UPDATE
DELETE
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.

'RECORDLENGTH'
Returns the record length in bytes.

'TRANSACTIONINDICATOR'
Returns the transaction indicator. The value corresponds to the TransInd field of the record header, which can be viewed with the Logdump utility.
Possible results are:

- BEGIN (represents TransInD of 0, the first record of a transaction.)
- MIDDLE (represents TransInD of 1, a record in the middle of a transaction.)
- END (represents TransInD of 2, the last record of a transaction.)
- WHOLE (represents TransInD of 3, the only record in a transaction.)

'GGFILEHEADER' , '*header_info*'

Valid for Replicat only.

Use the GGFILEHEADER option of @GETENV to return attributes of an Oracle GoldenGate Extract file or trail file. These attributes are stored as tokens in the file header.

> **Note:**
>
> If a given database, operating system, or Oracle GoldenGate version does not provide information that relates to a given token, a `NULL` value will be returned.

Syntax

```
@GETENV ('GGFILEHEADER', {'COMPATIBILITY'|'CHARSET'|'CREATETIMESTAMP'|
   'FILENAME'|'FILETYPE'|'FILESEQNO'|'FILESIZE'|'FIRSTRECCSN'|
   'LASTRECCSN'|'FIRSTRECIOTIME'|'LASTRECIOTIME'|'URI'|'URIHISTORY'|
   'GROUPNAME'|'DATASOURCE'|'GGMAJORVERSION'|'GGMINORVERSION'|
   'GGVERSIONSTRING'|'GGMAINTENANCELEVEL'|'GGBUGFIXLEVEL'|'GGBUILDNUMBER'|
   'HOSTNAME'|'OSVERSION'|'OSRELEASE'|'OSTYPE'|'HARDWARETYPE'|
   'DBNAME'|'DBINSTANCE'|'DBTYPE'|'DBCHARSET'|'DBMAJORVERSION'|
   'DBMINORVERSION'|'DBVERSIONSTRING'|'DBCLIENTCHARSET'|'DBCLIENTVERSIONSTRING'|
   'LASTCOMPLETECSN'|'LASTCOMPLETEXIDS'|'LASTCSN'|'LASTXID'|
   'LASTCSNTS'|'RECOVERYMODE'})
```

`'COMPATIBILITY'`
Returns the compatibility level of the trail file. The compatibility level of the current Oracle GoldenGate version must be greater than, or equal to, the compatibility level of the trail file to be able to read the data records in that file. Current valid values are from 0 or 6.

- 1 means that the trail file is of Oracle GoldenGate version 10.0 or later, which supports file headers that contain file versioning information.
- 0 means that the trail file is of an Oracle GoldenGate version that is older than 10.0. File headers are not supported in those releases. The 0 value is used for backward compatibility to those Oracle GoldenGate versions.
- 5 means that the trail file is of Oracle GoldenGate version 12.2 or later.
- 6 means that the trail file is of Oracle GoldenGate version 12.3.0.1.

  This value keeps increasing as per the Oracle GoldenGate version depending on the trail file version.

`'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.

`'FILENAME'`
Returns the name of the trail file. Can be an absolute or relative path, with a forward or backward slash depending on the file system.

`'FILETYPE'`
Returns a numerical value 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. The valid values are:

- 0 - EXTFILE

- 1 - EXTTRAIL
- 2 - UNIFIED and EXTFILE
- 3 - UNIFIED and EXTTRAIL

'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. For more information about the CSN, see *Administering Oracle GoldenGate*.

'LASTRECCSN'
Returns the commit sequence number (CSN) of the last record in the trail file.Value is `NULL` until the trail file is completed. For more information about the CSN, see *Administering Oracle GoldenGate*.

'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.

'RECOVERYMODE'
Returns recovery information for internal Oracle GoldenGate use. It is usually set to `APPENDMODE`.

'URI'
Returns the universal resource identifier of the process that created the trail file, in the following format:

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 Oracle GoldenGate installation path.
- `group_name` is the name of the process group that is linked with the process.

The following example shows where the trail was processed and by which process. This includes a history of previous runs.

```
sys1:home:oracle:v9.5:extora
```

'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.

`'GROUPNAME'`
Returns the name of the group that is associated with the Extract process that created the trail. The group name is the one that was supplied when the `ADD EXTRACT` command was issued.

`'DATASOURCE'`
Returns the data source that was read by the process as a number. The return value can be one of the following:

- `DS_EXTRACT_TRAILS`: The source was an Oracle GoldenGate extract file, populated with change data. The return value is `0`.
- `DS_DATABASE:` The source was a direct select from database table written to a trail, used for `SOURCEISTABLE`-driven initial load. The return value is `2`.
- `DS_TRAN_LOGS`: The source was the database transaction log. The return value is `3`.
- `DS_INITIAL_DATA_LOAD`: The source was a direct select from database tables for an initial load. The return value is `4`.
- `DS_VAM_EXTRACT:` The source was a vendor access module (VAM). The return value is `5`.
- `DS_VAM_TWO_PHASE_COMMIT:` The source was a VAM trail. The return value is `6`.

`'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.

`'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.

`'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:

- Version s10_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

'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

'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.

'DBTYPE'
Returns the type of database that produced the data in the trail file. Can be one of:

```
DB2 UDB
DB2 ZOS
MSSQL
MYSQL
ORACLE
TERADATA
ODBC
```

'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.)

'LASTCOMPLETECSN'
Returns recovery information for internal Oracle GoldenGate use.

'LASTCOMPLETEXIDS'
Returns recovery information for internal Oracle GoldenGate use.

'LASTCSN'
Returns recovery information for internal Oracle GoldenGate use.

'LASTXID'
Returns recovery information for internal Oracle GoldenGate use.

'LASTCSNTS'
Returns recovery information for internal Oracle GoldenGate use.

'RECORD' , '*location_info*'

Valid for a data pump Extract or Replicat.

Use the `RECORD` option of `@GETENV` to return the location or Oracle rowid of a record in an Oracle GoldenGate trail file.

Syntax

```
@GETENV ('RECORD',
{'TIMESTAMP_PRECISE'|'FILESEQNO'|'FILERBA'|'ROWID'|'RSN'|'TIMESTAMP'})
```

'TIMESTAMP_PRECISE'
Valid for a data pump, Extract, or Replicat.
The `TIMESTAMP_PRECISE` option returns the timestamp from year to microseconds. However, depending on the database, the value can be in milliseconds with 0 microseconds.

'FILESEQNO'
Returns the sequence number of the trail file without any leading zeros.

'FILERBA'
Returns the relative byte address of the record within the FILESEQNO file.

'ROWID'
(Valid for Oracle) Returns the row id of the record.

'RSN'
Returns the record sequence number within the transaction.

'TIMESTAMP'
Returns the timestamp of the record.

Example:

```
REC-TIMESTAMP: 2017-10-31 06:21:07 REC-TIMESTAMP-PRECISE: 2017-10-31
06:21:07.478064
```

'DBENVIRONMENT' , '*database_info*'

Valid for Extract and Replicat.

Use the DBENVIRONMENT option of @GETENV to return global environment information for a database.

Syntax

```
@GETENV ('DBENVIRONMENT', {'DBNAME'|'DBVERSION'|'DBUSER'|'SERVERNAME'})
```

'DBNAME'
Returns the database name.

'DBVERSION'
Returns the database version.

'DBUSER'
Returns the database login user. Note that SQL Server does not log the user ID.

'SERVERNAME'
Returns the name of the server.

'TRANSACTION' , '*transaction_info*

Valid for Extract.

Use the TRANSACTION option of @GETENV to return information about a source transaction. This option is valid for the Extract process but not for pump Extract and Replicat.

Syntax

```
@GETENV ('TRANSACTION',
{'TIMESTAMP_PRECISE'|'TRANSACTIONID'|'XID'|'CSN'|'TIMESTAMP'|'NAME'|
   'USERID'|'USERNAME'|'PLANNAME' | 'LOGBSN' | 'REDOTHREAD' | 'PROGRAMNAME' |
'CLIENTIDENTIFIER' | 'MACHINENAME' | 'USERNAME')
```

> **Note:**
>
> Do not use TIMETSAMP_PRECISE or TIMESTAMP for ordering operations. Instead use this value: @COMPUTE(@COMPUTE(@NUMSTR(@GETENV ("RECORD", "FILESEQNO"))*100000000000)+@NUMSTR(@GETENV ("RECORD", "FILERBA")))

'TIMESTAMP_PRECISE'
This option is valid for Extract. Use the TIMESTAMP_PRECISE returns the timestamp from year to microseconds. However, depending on the database, the value can be in milliseconds with 0 microseconds

'TRANSACTIONID' | 'XID'
Returns the transaction ID number. Either TRANSACTIONID or XID can be used. The transaction ID and the CSN are associated with the first record of every transaction and are stored as tokens in the trail record. For each transaction ID, there is an associated CSN. Transaction ID tokens have no zero-padding on any platform, because they never get evaluated as relative values. They only get evaluated for whether they match or do not match. Note that in the trail, the transaction ID token is shown as TRANID.

'CSN'
Returns the commit sequence number (CSN). The CSN is not zero-padded when returned for these databases: Oracle, DB2 LUW, and DB2 z/OS. For all other supported databases, the CSN is zero-padded.
Note that in the trail, the CSN token is shown as LOGCSN. See the TRANSACTIONID | XID environment value for additional information about the CSN token.
For more information about the CSN, see *Administering Oracle GoldenGate*.

'TIMESTAMP'
Returns the commit timestamp of the transaction.

'NAME'
Returns the transaction name, if available.

'USERID'
(Oracle) Returns the Oracle user ID of the database user that committed the last transaction. This is not valid for pump Extract and/or Replicat.

'USERNAME'
(Oracle) Returns the Oracle user name of the database user that committed the last transaction. This is not valid for pump Extract and/or Replicat.

'PLANNAME'
(DB2 z/OS) Returns the plan name under which the current transaction was originally executed. The plan name is included in the begin unit of recovery log record.

'LOGBSN'
Returns the begin sequence number (BSN) in the transaction log. The BSN is the native sequence number that identifies the beginning of the oldest uncommitted transaction that is held in Extract memory. For example, given an Oracle database, the BSN would be expressed as a system change number (SCN). The BSN corresponds to the current I/O checkpoint value of Extract. This value can be obtained

from the trail by Replicat when @GETENV ('TRANSACTION', 'LOGBSN') is used. This value also can be obtained by using the INFO REPLICAT command with the DETAIL option. The purpose of obtaining the BSN from Replicat is to get a recovery point for Extract in the event that a system failure or file system corruption makes the Extract checkpoint file unusable. See *Administering Oracle GoldenGate* for more information about recovering the Extract position.

'REDOTHREAD'
Returns the thread number of a RAC node extract; on non-RAC node extracts the value is always 1. For data pump and Replicat, the thread id used by Extract capture of a RAC node is returned; on non-RAC, @GETENV() returns an error. Logdump shows the token, ORATHREADID, in the token section if the transaction is captured by Extract on a RAC node.

'PROGRAMNAME'
Name of the program or application that started the transaction or session.

'CLIENTIDENTIFIER'
Value set by using DBMS_SESSION_.set_identifier().

'MACHINENAME'
Name of the host, machine, or server where database is running

'USERNAME'
Database login user name.

Example:

```
DB2 zOS:
TRANS-TIMESTAMP:          2017-10-31 06:21:07
TRANS-TIMESTAMP-PRECISE: 2017-10-31 06:21:07.485792
```

'OSVARIABLE' , '*variable*'

Valid for Extract and Replicat.

Use the OSVARIABLE option of @GETENV to return the string value of a specified operating-system environment variable.

Syntax

```
@GETENV ('OSVARIABLE', 'variable')
```

'variable'
The name of the variable. The search is an exact match of the supplied variable name. For example, the UNIX grep command would return all of the following variables, but @GETENV ('OSVARIABLE', 'HOME') would only return the value for HOME:

```
ANT_HOME=/usr/local/ant
JAVA_HOME=/usr/java/j2sdk1.4.2_10
HOME=/home/judyd
ORACLE_HOME=/rdbms/oracle/ora1022i/64
```

The search is case-sensitive if the operating system supports case-sensitivity.

'TLFKEY' , SYSKEY, '*unique_key*'

Valid for Extract and Replicat.

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).

Syntax

```
@GETENV ('TLFKEY', SYSKEY, unique_key)
```

SYSKEY, *unique_key*
The NonStop node number of the source TLF/PTLF file. Do not enclose this syntax element in quotes.
Example:

```
GETENV ('TLFKEY', SYSKEY, 27)
```

4.24 GET_ENV_VALUE

Valid For

Extract and Replicat

Description

Use the GET_ENV_VALUE function to return information about the Oracle GoldenGate environment. The information that is supplied is the same as that of the @GETENV column-conversion function and is specified by using the same input values. For more information about the valid information types, environment variables, and return values, see "GETENV".

If the character session of the user exit is set with SET_SESSION_CHARSET to a value other than the default character set of the operating system, as defined in ULIB_CS_DEFAULT in the ucharset.h file, the character data that is exchanged between the user exit and the process is interpreted in the session character set.

Syntax

```
#include "usrdecs.h"
short result_code;
getenv_value_def env_ptr;
ERCALLBACK (GET_ENV_VALUE, &env_ptr, &result_code);
```

Buffer

```
typedef struct
{
char *information_type;
char *env_value_name;
char *return_value;
long max_return_length;
long actual_length;
short value_truncated;
} getenv_value_def;
```

Input

`information_type`
The information type that is to be returned, for example `'GGENVIRONMENT'` or `'GGHEADER'`. The information type must be supplied within double quotes. For a list of information types and subsequent detailed descriptions, see "GETENV".

`env_value_name`
The environment value that is wanted from the information type. The environment value must be supplied within double quotes. For valid values, see "GETENV". For example, if using the `'GGENVIRONMENT'` information type, a valid environment value would be `'GROUPNAME'`.

`max_return_length`
The maximum length of the buffer for this data.

Output

`return_value`
A valid return value for the supplied environment value.

`actual_length`
The actual length of the data in this buffer.

`value_truncated`
A flag (`0` or `1`) to indicate whether or not the value was truncated. Truncation occurs if the length of the value plus the null terminator exceeds the maximum buffer length.

Return Values

```
EXIT_FN_RET_OK
EXIT_FN_RET_ENV_NOT_FOUND
EXIT_FN_RET_INVALID_PARAM
```

4.25 GET_ERROR_INFO

Valid For

Extract and Replicat

Description

Use the `GET_ERROR_INFO` function to retrieve error information associated with a discard record. The user exit can use this information in custom error handling logic. For example, the user exit could send an e-mail message with detailed error information.

If the character session of the user exit is set with `SET_SESSION_CHARSET` to a value other than the default character set of the operating system, as defined in `ULIB_CS_DEFAULT` in the `ucharset.h` file, the message data that is exchanged between the user exit and the process is interpreted in the session character set.

Syntax

```
#include "usrdecs.h"
short result_code;
```

```
error_info_def error_info;
ERCALLBACK (GET_ERROR_INFO, &error_info, &result_code);
```

Buffer

```
typedef struct
{
long error_num;
char *error_msg;
long max_length;
long actual_length;
short msg_truncated;
} error_info_def;
```

Input

error_msg
A pointer to a buffer to accept the returned error message.

max_length
The maximum length of your allocated `error_msg` buffer to accept any resulting error message. This is returned as a `NULL` terminated string.

Output

error_num
The SQL or system error number associated with the discarded record.

error_msg
A pointer to the null-terminated error message string associated with the discarded record.

actual_length
The length of the error message, not including the null terminator.

msg_truncated
A flag (`0` or `1`) indicating whether or not the error message was truncated. Truncation occurs if the length of the error message plus a null terminator exceeds the maximum buffer length.

Return Values

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_OK
```

4.26 GET_GMT_TIMESTAMP

Valid For

Extract and Replicat

Description

Use the `GET_GMT_TIMESTAMP` function to retrieve the operation commit timestamp in GMT format. This function requires compiling with Version 2 `usrdecs.h` or later.

Syntax

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (GET_GMT_TIMESTAMP, &record, &result_code);
```

Buffer

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

Input

None

Output

`timestamp`
The returned 64-bit I/O timestamp in GMT format.

`io_datetime`
A null-terminated string containing the local I/O date and time:
`YYYY-MM-DD HH:MI:SS.FFFFFF`
The format of the datetime string is in the session character set.

Return Values

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_OK
```

4.27 GET_MARKER_INFO

Valid For

Extract (data pump only) and Replicat

Description

Use the `GET_MARKER_INFO` function to retrieve marker information sent from a NonStop source system when Replicat is applying data. Use markers to trigger custom processing within a user exit.

If the character session of the user exit is set with SET_SESSION_CHARSET to a value other than the default character set of the operating system, as defined in ULIB_CS_DEFAULT in the ucharset.h file, all of the returned marker data is interpreted in the session character set.

Syntax

```
#include "usrdecs.h"
short result_code;
marker_info_def marker_info;
ERCALLBACK (GET_MARKER_INFO, &marker_info, &result_code);
```

Buffer

```
typedef struct
{
char *processed;
char *added;
char *text;
char *group;
char *program;
char *node;
} marker_info_def;
```

Input

processed
A pointer to a buffer to accept the processed return value.

added
A pointer to a buffer to accept the added return value.

text
A pointer to a buffer to accept the text return value.

group
A pointer to a buffer to accept the group return value.

program
A pointer to a buffer to accept the program return value.

node
A pointer to a buffer to accept the node return value.

Output

processed
A null-terminated string in the format of YYYY-MM-DD HH:MI:SS indicating the local date and time that the marker was processed.

added
A null-terminated string in the format of YYYY-MM-DD HH:MI:SS indicating the local date and time that the marker was added.

text
A null-terminated string containing the text associated with the marker.

group
A null-terminated string indicating the Replicat group that processed the marker.

program
A null-terminated string indicating the program that processed the marker.

node
A null-terminated string representing the Himalaya node on which the marker was originated.

Return Values

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_OK
```

4.28 GET_OBJECT_NAME

Valid For

Extract and Replicat

Description

Use the GET_OBJECT_NAME function to retrieve the fully qualified name of a source or target object that is associated with the record being processed. This function is valid tables and other objects in a DML or DDL operation.

To return only part of the object name, see the following:

GET_OBJECT_NAME_ONLY GET_SCHEMA_NAME_ONLY GET_CATALOG_NAME_ONLY

Database object names are returned exactly as they are defined in the hosting database, including the letter case.

Syntax

```
#include "usrdecs.h"
short result_code;
env_value_def env_value;
ERCALLBACK (GET_OBJECT_NAME, &env_value, &result_code);
```

Buffer

```
typedef struct
{
char *buffer;
long max_length;
long actual_length;
short value_truncated;
short index;
short source_or_target;
} env_value_def;
```

Input

buffer
A pointer to a buffer to accept the returned object name. The name is null-terminated.

max_length
The maximum length of your allocated buffer to accept the object name. This is returned as a NULL terminated string.

source_or_target
One of the following indicating whether to return the source or target object name.

EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL

Output

buffer
The fully qualified, null-terminated object name, for example schema.object or catalog.schema.object, depending on the database platform.
If the character session of the user exit is set with SET_SESSION_CHARSET to a value other than the default character set of the operating system, as defined in ULIB_CS_DEFAULT in the ucharset.h file, the object name is interpreted in the session character set.

actual length
The string length of the returned object name. The actual length does not include the null terminator. The actual length is 0 if the object is a table.

value_truncated
A flag (0 or 1) indicating whether or not the value was truncated. Truncation occurs if the length of the object name plus the null terminator exceeds the maximum buffer length.

Return Values

EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK

4.29 GET_OBJECT_NAME_ONLY

Valid For

Extract and Replicat

Description

Use the GET_OBJECT_NAME_ONLY function to retrieve the unqualified name (without the catalog, container, or schema) of a source or target object that is associated with the record that is being processed. This function is valid for tables and other objects in a DML or DDL operation.

To return the fully qualified name of an object, see the following:

GET_OBJECT_NAME

To return other parts of the object name, see the following:

GET_SCHEMA_NAME_ONLY GET_CATALOG_NAME_ONLY

Database object names are returned exactly as they are defined in the hosting database, including the letter case.

Syntax

```
#include "usrdecs.h"
short result_code;
env_value_def env_value;
ERCALLBACK (GET_OBJECT_NAME_ONLY, &env_value, &result_code);
```

Buffer

```
typedef struct
{
char *buffer;
long max_length;
long actual_length;
short value_truncated;
short index;
short source_or_target;
} env_value_def;
```

Input

`buffer`
A pointer to a buffer to accept the returned object name. The name is null-terminated.

`max_length`
The maximum length of your allocated buffer to accept the object name. This is returned as a `NULL` terminated string.

`source_or_target`
One of the following indicating whether to return the source or target object name.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

`buffer`
The fully qualified, null-terminated object name, for example `schema.object` or `catalog.schema.object`, depending on the database platform.
If the character session of the user exit is set with `SET_SESSION_CHARSET` to a value other than the default character set of the operating system, as defined in `ULIB_CS_DEFAULT` in the `ucharset.h` file, the object name is interpreted in the session character set.

`actual length`
The string length of the returned object name. The actual length does not include the null terminator. The actual length is 0 if the object is a table.

`value_truncated`
A flag (`0` or `1`) indicating whether or not the value was truncated. Truncation occurs if the length of the object name plus the null terminator exceeds the maximum buffer length.

Return Values

```
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

4.30 GET_OPERATION_TYPE

Valid For

Extract and Replicat

Description

Use the GET_OPERATION_TYPE function to determine the operation type associated with a record. Knowing the operation type can be useful in a user exit. For example, the user exit can perform complex validations any time a delete is encountered. It also is important to know when a compressed record is being processed if the user exit is manipulating the full data record.

As an alternative, you can use the GET_RECORD_BUFFER function to determine the operation type (see "GET_RECORD_BUFFER").

Syntax

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (GET_OPERATION_TYPE, &record, &result_code);
```

Buffer

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

Input

source_or_target
One of the following indicating whether to return the operation type for the source or the target data record.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

`io_type`
Returned as one of the following:

- **DDL type:**

  ```
  SQL_DDL_VAL
  ```

- **DML types:**

  ```
  DELETE_VAL
  INSERT_VAL
  UPDATE_VAL
  ```

- **Compressed SQL update:**

  ```
  UPDATE_COMP_SQL_VAL
  UPDATE_COMP_PK_SQL_VAL
  ```

- **Other:**

  ```
  TRUNCATE_TABLE_VAL
  ```

Return Values

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

4.31 GET_POSITION

Valid For

Extract (data pump only) and Replicat

Description

Use the `GET_POSITION` function is obtain a read position of an Extract data pump or Replicat in the Oracle GoldenGate trail.

Syntax

```
#include "usrdecs.h"
short result_code;
ERCALLBACK (GET_POSITION &position_def, &result_code);
```

Buffer

```
typedef struct
{
char *position;
long position_len;
short position_type;
short ascii_or_internal;
} position_def;
```

Input

position_len
Allocation length for the position length.

position_type
Can be one of the following:

STARTUP_CHECKPOINT
The start position in the trail.

CURRENT_CHECKPOINT
The position of the last read in the trail.

column_value_mode
An indicator for the format in which the column value was passed. Currently, only the default Oracle GoldenGate canonical format is supported, as represented by:
EXIT_FN_RAW_FORMAT

Output

***position**
A pointer to a buffer representing the position values. This buffer is declared in the position_def as two binary values (unsigned int32t and int32t) as seqnorba for eight bytes in a char field. The user exit must move the data to the correct data type. Using this function on a Little Endian platform will cause the process to "reverse bytes" on the two fields individually.

Return Values

EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_NOT_SUPPORTED
EXIT_FN_RET_OK

4.32 GET_RECORD_BUFFER

Valid For

Extract and Replicat

Description

Use the GET_RECORD_BUFFER function to obtain information for custom column conversions. User exits can be used for data mapping between dissimilar source and target records when the COLMAP option of the MAP or TABLE parameter is not sufficient. For example, you can use a user exit to convert a proprietary date field.

You can use the SET_RECORD_BUFFER function (see "SET_RECORD_BUFFER") to modify the data retrieved with GET_RECORD_BUFFER. However, it requires an understanding of the data record as written in the internal Oracle GoldenGate canonical format. As an alternative, you can set column values in the data record with the SET_COLUMN_VALUE_BY_INDEX function (see "SET_COLUMN_VALUE_BY_INDEX") or the SET_COLUMN_VALUE_BY_NAME function (see "STRNCMP").

Deletes, inserts and updates appear in the buffer as full record images.

Compressed SQL updates have the following 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 one of the following two-byte null indicators when applicable. `0` is not null. `-1` is null.

For SQL records, you can use the `DECOMPRESS_RECORD` function ("DECOMPRESS_RECORD") to decompress the record for possible manipulation and then use the `COMPRESS_RECORD` function ("COMPRESS_RECORD") to compress it again, as expected by the process.

Syntax

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (GET_RECORD_BUFFER, &record, &result_code);
```

Buffer

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

Input

`source_or_target`
One of the following indicating whether to return the record buffer for the source or target data record.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

`requesting_before_after_ind`
Optional. Set when requesting a record buffer on a record io_type of `UPDATE_COMP_PK_SQL_VAL` (primary key update). Use one of the following to indicate which portion of the primary key update is to be accessed. The default is `AFTER_IMAGE_VAL`.

```
BEFORE_IMAGE_VAL
AFTER_IMAGE_VAL
```

Output

buffer
A pointer to the record buffer. Typically, buffer is a pointer to a buffer of type `exit_rec_buf_def`. The `exit_rec_buf_def` buffer contains the actual record about to be processed by Extract or Replicat. The buffer is supplied when the call type is `EXIT_CALL_DISCARD_RECORD`. Exit routines can change the contents of this buffer, for example, to perform custom mapping functions.
The content of the record buffer is not converted to or from the character set of the user exit. It is passed as-is.

length
The returned length of the record buffer.

io_type
Returned as one of the following:

- **DDL type:**

  ```
  SQL_DDL_VAL
  ```

- **DML types:**

  ```
  DELETE_VAL
  INSERT_VAL
  UPDATE_VAL
  ```

- **Compressed SQL update:**

  ```
  UPDATE_COMP_SQL_VAL
  UPDATE_COMP_PK_SQL_VAL
  ```

- **Other:**

  ```
  TRUNCATE_TABLE_VAL
  ```

mapped
A flag (`0` or `1`) indicating whether or not this is a mapped record buffer.

before_after_ind
One of the following to indicate whether the record is a before or after image.

```
BEFORE_IMAGE_VAL
AFTER_IMAGE_VAL
```

Return Values

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

4.33 GET_RECORD_LENGTH

Valid For

Extract and Replicat

Description

Use the GET_RECORD_LENGTH function to retrieve the length of the data record. As an alternative, you can use the GET_RECORD_BUFFER function to retrieve the length of the data record.

Syntax

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (GET_RECORD_LENGTH, &record, &result_code);
```

Buffer

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

Input

source_or_target
One of the following indicating whether to return the record length for the source or target data record.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

length
The returned length of the data record.

Return Values

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

4.34 GET_RECORD_TYPE

Valid For

Extract and Replicat

Description

Use the GET_RECORD_TYPE function to retrieve the type of record being processed. The record can be a SQL record. The record type is important when manipulating the record buffer, because each record type has a different format.

Syntax

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (GET_RECORD_TYPE, &record, &result_code);
```

Buffer

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

Input

source_or_target
One of the following indicating whether or not to return the record type for the source or target data record.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

record_type
The returned record type.

```
EXIT_REC_TYPE_SQL
```

Return Values

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

4.35 GET_SCHEMA_NAME_ONLY

Valid For

Extract and Replicat

Description

Use the GET_SCHEMA_NAME_ONLY function to retrieve the name of the owner (such as schema), but not the name of the catalog or container (if applicable) or the object, of the source or target object associated with the record being processed. This function is valid for DML and DDL operations.

To return the fully qualified name of a table, see the following:

GET_TABLE_NAME

To return the fully qualified name of a non-table object, such as a user, view or index, see the following:

GET_OBJECT_NAME

To return only the unqualified table or object name, see the following:

GET_TABLE_NAME_ONLY

GET_OBJECT_NAME_ONLY

To return other parts of the table or object name, see the following:

GET_CATALOG_NAME_ONLY

Database object names are returned exactly as they are defined in the hosting database, including the letter case.

Syntax

```
#include "usrdecs.h"
short result_code;
env_value_def env_value;
ERCALLBACK (GET_SCHEMA_NAME_ONLY, &env_value, &result_code);
```

Buffer

```
typedef struct
{
char *buffer;
long max_length;
long actual_length;
short value_truncated;
short index;
short source_or_target;
} env_value_def;
```

Input

buffer
A pointer to a buffer to accept the returned schema name. The name is null-terminated.

max_length
The maximum length of your allocated buffer to accept the schema name. This is returned as a NULL terminated string.

source_or_target
One of the following indicating whether to return the source or target schema name.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

buffer
The fully qualified, null-terminated schema name.
If the character session of the user exit is set with SET_SESSION_CHARSET to a value other than the default character set of the operating system, as defined in ULIB_CS_DEFAULT in the ucharset.h file, the schema name is interpreted in the session character set.

actual_length
The string length of the returned name. The actual length does not include the null terminator.

value_truncated
A flag (0 or 1) indicating whether or not the value was truncated. Truncation occurs if the length of the schema name plus the null terminator exceeds the maximum buffer length.

Return Values

```
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

4.36 GET_SESSION_CHARSET

Valid For

Extract and Replicat

Description

Use GET_SESSION_CHARSET to get the current user exit session character set. This character set can be set through callback function SET_SESSION_CHARSET. The character set of the user exit session indicates the encoding of any character-based callback structure members that are used between the user exit and the caller process (Extract, data pump, Replicat), including metadata such as (but not limited to):

- database names and locales

- table and column names
- DDL text
- error messages
- character-type columns such as `CHAR` and `NCHAR`
- date-time and numeric columns that are represented in string form

The valid values of the session character set are defined in the header file `ucharset.h`. This function can be called at any time that the user exit has control.

For more information about globalization support, see Administering Oracle GoldenGate for Windows and UNIX.

Syntax

```
#include usrdecs.h
short result_code;
session_def session_charset_def;
ERCALLBACK (GET_SESSION_CHARSET, &session_charset_def, &result_code);
```

Buffer

```
typedef struct
{
ULibCharSet  session_charset;
} session_def;
```

Input

None

Output

```
session_charset_def.session_charset
```

Return Values

```
EXIT_FN_RET_OK
```

4.37 GET_STATISTICS

Valid For

Extract and Replicat

Description

Use the `GET_STATISTICS` function to retrieve the current processing statistics for the Extract or Replicat process. For example, the user exit can output statistics to a custom report should a fatal error occur during Extract or Replicat processing.

Statistics are automatically handled based on which process type has requested the data:

- The Extract process will always treat the request as a source table, counting that table once regardless of the number of times output.

- The Replicat process will always treat the request as a set of target tables. The set includes all counts to the target regardless of the number of source tables.

If the database is case-sensitive, object names must be specified in the same letter case as they are defined in the hosting database; otherwise, the case does not matter.

Syntax

```
#include "usrdecs.h"
short result_code;
statistics_def statistics;
ERCALLBACK (GET_STATISTICS, &statistics, &result_code);
```

Buffer

```
typedef struct
{
char *table_name;
short group;
exit_timestamp_string start_datetime;
long num_inserts;
long num_updates;
long num_befores;
long num_deletes;
long num_discards;
long num_ignores;
long total_db_operations;
long total_operations;
/* Version 2 CALLBACK_STRUCT_VERSION */
long num_truncates;
} statistics_def;
```

Input

`table_name`
A null-terminated string specifying the fully qualified name of the source table. Statistics are always recorded against the source records. If the character session of the user exit is set with `SET_SESSION_CHARSET` to a value other than the default character set of the operating system, as defined in `ULIB_CS_DEFAULT` in the `ucharset.h` file, the table name and the date are interpreted in the session character set.

`group`
Can be one of the following:

`EXIT_STAT_GROUP_STARTUP`
Retrieves statistics since the Oracle GoldenGate process was last started.

`EXIT_STAT_GROUP_DAILY`
Retrieves statistics since midnight of the current day.

`EXIT_STAT_GROUP_HOURLY`
Retrieves statistics since the start of the current hour.

`EXIT_STAT_GROUP_RECENT`
Retrieves statistics since the statistics were reset using GGSCI.

EXIT_STAT_GROUP_REPORT
Retrieves statistics since the last report was generated.

EXIT_STAT_GROUP_USEREXIT
Retrieves statistics since the last time the user exit reset the statistics with RESET_USEREXIT_STATS.

Output

start_datetime
A null-terminated string in the format of YYYY-MM-DD HH:MI:SS indicating the local date and time that statistics started to be recorded for the specified group.

num_inserts
The returned number of inserts processed by Extract or Replicat.

num_updates
The returned number of updates processed by Extract or Replicat.

num_befores
The returned number of update before images processed by Extract or Replicat.

num_deletes
The returned number of deletes processed by Extract or Replicat.

num_discards
The returned number of records discarded by Extract or Replicat.

num_ignores
The returned number of records ignored by Extract or Replicat.

total_db_operations
The returned number of total database operations processed by Extract or Replicat.

total_operations
The returned number of total operations processed by Extract or Replicat, including discards and ignores.

num_truncates
The returned number of truncates processed by Extract or Replicat.

Return Values

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_TABLE_NOT_FOUND
EXIT_FN_RET_OK
```

4.38 GET_TABLE_COLUMN_COUNT

Valid For

Extract and Replicat

Description

Use the GET_TABLE_COLUMN_COUNT function to retrieve the total number of columns in a table, including the number of key columns.

Syntax

```
#include "usrdecs.h"
short result_code;
table_def table;
ERCALLBACK (GET_TABLE_COLUMN_COUNT, &table, &result_code);
```

Buffer

```
typedef struct
{
short num_columns;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION */
short num_key_columns;
} table_def;
```

Input

source_or_target
One of the following indicating whether to return the total number of columns for the source or target table.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

num_columns
The returned total number of columns in the specified table.

num_key_columns
The returned total number of columns that are being used by Oracle GoldenGate as the key for the specified table.

Return Values

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

4.39 GET_TABLE_METADATA

Valid For

Extract and Replicat

Description

Use the GET_TABLE_METADATA function to retrieve metadata about the table that associated with the record that is being processed.

Syntax

```
#include "usrdecs.h"
short result_code;
table_metadata_def tbl_meta_rec;
ERCALLBACK (GET_TABLE_METADATA, &tbl_meta_rec, &result_code);
```

Buffer

```
typedef struct
{
char *table_name;
short value_truncated;
long max_name_length;
long actual_name_length;
short num_columns;
short num_key_columns;
short *key_columns;
short num_keys_returned;
BOOL using_pseudo_key;
short source_or_target;
} table_metadata_def;
```

Input

`table_name`
A pointer to a buffer to accept the `table_name` return value

`key_columns`
A pointer to an array of `key_columns` indexes.

`max_name_length`
The maximum length of the returned table name. Typically, the maximum length is the length of the table name buffer. Since the returned table name is null-terminated, the maximum length should equal the maximum length of the table name.

`source_or_target`
One of the following indicating whether to return the source or target table name.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

`table_name`
The name of the table associated with the record that is being processed. If the character session of the user exit is set with `SET_SESSION_CHARSET` to a value other than the default character set of the operating system, as defined in `ULIB_CS_DEFAULT` in the `ucharset.h` file, the table name is interpreted in the session character set.

`value_truncated`
A flag (`0` or `1`) indicating whether or not the value was truncated. Truncation occurs if the length of the table name plus the null terminator exceeds the maximum buffer length.

actual_name_length
The string length of the returned table name. The actual length does not include the null terminator.

num_columns
The number of columns in the table.

num_key_columns
The number of columns in the key that is being used by Oracle GoldenGate.

key_columns
The values for the key columns. You must know the expected number of keys multiplied by the length of the columns, and then allocate the appropriate amount of buffer.

num_keys_returned
The number of key columns that are requested.

using_pseudo_key
A flag that indicates whether or not KEYCOLS-specified columns are being used as a key. Returns TRUE or FALSE.

Return Values

```
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_EXCEEDED_MAX_LENGTH
EXIT_FN_RET_OK
```

4.40 GET_TABLE_NAME

Valid For

Extract and Replicat

Description

Use the GET_TABLE_NAME function to retrieve the fully qualified name of the source or target table associated with the record being processed. This function is valid only for tables in DML and DDL operations. To retrieve the fully qualified name of a non-table object, see the following:

GET_OBJECT_NAME

To return only part of the fully qualified name, see also the following:

GET_TABLE_NAME_ONLY GET_SCHEMA_NAME_ONLY GET_CATALOG_NAME_ONLY

Database object names are returned exactly as they are defined in the hosting database, including the letter case.

This function returns a value only if the object is a table. Otherwise, the actual_length of the env_value_def variable returns 0.

Syntax

```
#include "usrdecs.h"
short result_code;
```

```
env_value_def env_value;
ERCALLBACK (GET_TABLE_NAME, &env_value, &result_code);
```

Buffer

```
typedef struct
{
char *buffer;
long max_length;
long actual_length;
short value_truncated;
short index;
short source_or_target;
} env_value_def;
```

Input

buffer
A pointer to a buffer to accept the returned table name. The table name is null-terminated.

max_length
The maximum length of your allocated buffer to accept the table name. This is returned as a NULL terminated string.

source_or_target
One of the following indicating whether to return the source or target table name.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

buffer
The fully qualified, null-terminated table name, for example schema.table or catalog.schema.table, depending on the database platform.
If the character session of the user exit is set with SET_SESSION_CHARSET to a value other than the default character set of the operating system, as defined in ULIB_CS_DEFAULT in the ucharset.h file, the table name is interpreted in the session character set.

actual length
The string length of the returned table name. The actual length does not include the null terminator. The actual length returned is 0 if the object is anything other than a table.

value_truncated
A flag (0 or 1) indicating whether or not the value was truncated. Truncation occurs if the length of the table name plus the null terminator exceeds the maximum buffer length.

Return Values

```
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

4.41 GET_TABLE_NAME_ONLY

Valid For

Extract and Replicat

Description

Use the GET_TABLE_NAME_ONLY function to retrieve the unqualified name (without the catalog, container, or schema) of the source or target table associated with the record being processed. This function is valid only for tables in DML and DDL operations. To retrieve the unqualified name of a non-table object, see the following:

GET_OBJECT_NAME_ONLY

To return the fully qualified name of a table, see the following:

GET_TABLE_NAME

To return other parts of the table name, see the following:

GET_SCHEMA_NAME_ONLY GET_CATALOG_NAME_ONLY

Database object names are returned exactly as they are defined in the hosting database, including the letter case.

This function returns a value only if the object is a table. Otherwise, the actual_length of the env_value_def variable returns 0.

Syntax

```
#include "usrdecs.h"
short result_code;
env_value_def env_value;
ERCALLBACK (GET_TABLE_NAME_ONLY, &env_value, &result_code);
```

Buffer

```
typedef struct
{
char *buffer;
long max_length;
long actual_length;
short value_truncated;
short index;
short source_or_target;
} env_value_def;
```

Input

buffer
A pointer to a buffer to accept the returned table name. The table name is null-terminated.

max_length
The maximum length of your allocated buffer to accept the table name. This is returned as a NULL terminated string.

source_or_target
One of the following indicating whether to return the source or target table name.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

buffer
The fully qualified, null-terminated table name, for example `schema.table` or `catalog.schema.table`, depending on the database platform.
If the character session of the user exit is set with `SET_SESSION_CHARSET` to a value other than the default character set of the operating system, as defined in `ULIB_CS_DEFAULT` in the `ucharset.h` file, the table name is interpreted in the session character set.

actual length
The string length of the returned table name. The actual length does not include the null terminator. The actual length returned is 0 if the object is anything other than a table.

value_truncated
A flag (`0` or `1`) indicating whether or not the value was truncated. Truncation occurs if the length of the table name plus the null terminator exceeds the maximum buffer length.

Return Values

```
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

4.42 GET_TIMESTAMP

Valid For

Extract and Replicat

Description

Use the `GET_TIMESTAMP` function to retrieve the I/O timestamp associated with a source data record in ASCII datetime format. The timestamp is then converted to local time and approximates the time of the original database operation.

> **Note:**
>
> The ASCII commit timestamp can vary with the varying regional use of Daylight Savings Time. The user exit callback should return the ASCII datetime as a GMT time to avoid this variance. The Oracle GoldenGate trail uses GMT format. See "GET_GMT_TIMESTAMP".

Syntax

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (GET_TIMESTAMP, &record, &result_code);
```

Buffer

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

Input

None

Output

`timestamp`
The returned 64-bit I/O timestamp in ASCII format.

`io_datetime`
A null-terminated string containing the local I/O date and time, in the format of:
`YYYY-MM-DD HH:MI:SS.FFFFFF`

Return Values

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_OK
```

4.43 GET_TRANSACTION_IND

Valid For

Extract and Replicat

Description

Use the `GET_TRANSACTION_IND` function to determine whether a data record is the first, last or middle operation in a transaction. This can be useful when, for example, a user exit can compile the details of each transaction and output a special summary record.

Syntax

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (GET_TRANSACTION_IND, &record, &result_code);
```

Buffer

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

Input

None

Output

`transaction_ind`
The returned transaction indicator, represented as one of the following:

`BEGIN_TRANS_VAL`
The record is the beginning of a transaction.

`MIDDLE_TRANS_VAL`
The record is in the middle of a transaction.

`END_TRANS_VAL`
The record is the end of a transaction.

`WHOLE_TRANS_VAL`
The record is the only one in the transaction.

Return Values

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_OK
```

4.44 GET_USER_TOKEN_VALUE

Valid For

Extract and Replicat

Description

Use the GET_USER_TOKEN_VALUE function to obtain the value of a user token from a trail record. No character-set conversion is performed on the token value.

Syntax

```
#include "usrdecs.h"
```

Buffer

```
typedef struct
{
char *token_name;
char *token_value;
long max_length;
long actual_length;
short value_truncated;
} token_value_def;
```

Input

token_name
A pointer to a buffer representing the name of a token. It is assumed that the token name is encoded in the default character set of the operating system that hosts the Extract TABLE statement where the token is configured. The user exit prepares the token name in the character set that is specified with SET_SESSION_CHARSET, but converts it back to the operating system character set before retrieving the matching token value.

max_length
The maximum length of your allocated token_name buffer to accept any resulting token value. This is returned as a NULL terminated string.

Output

token_value
A pointer to a buffer representing the return value (if any) of a token. The token value is passed back to the user exit as-is, without any character-set conversion.

actual_length
The actual length of the token value that is returned. A value of 0 is returned if the token is found and there is no value present.

value_truncated
A flag of either 0 or 1 that indicates whether or not the token value was truncated. Truncation occurs if the length of the table name plus the null terminator exceeds the maximum buffer length.

Return Values

```
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_TOKEN_NOT_FOUND
EXIT_FN_RET_OK
```

4.45 OUTPUT_MESSAGE_TO_REPORT

Valid For

Extract and Replicat

Description

Use the `OUTPUT_MESSAGE_TO_REPORT` function to output a message to the report file. If a character session for the user exit is set with `SET_SESSION_CHARSET`, the message is interpreted in the session character set but is converted to the default character set of the operating system before being written to the report file.

Syntax

```
#include "usrdecs.h"
short result_code;
char  message[500];
ERCALLBACK (OUTPUT_MESSAGE_TO_REPORT, message, &result_code);
```

Buffer

None

Input

`message`
A null-terminated string.

Output

None

Return Values

`EXIT_FN_RET_OK`

4.46 RESET_USEREXIT_STATS

Valid For

Extract and Replicat

Description

Use the `RESET_USEREXIT_STATS` function to reset the `EXIT_STAT_GROUP_USEREXIT` statistics for the Oracle GoldenGate process since the last call to `GET_STATISTICS` was processed. This function enables the user exit to control when to reset the group statistics that are returned by the `GET_STATISTICS` function, but does not permit any of the other statistics to be reset.

Syntax

```
#include "usrdecs.h"
short result_code;
call_callback (RESET_USEREXIT_STATS, NULL, &result_code);
```

Input

None

Output

None

Return Values

None

4.47 SET_COLUMN_VALUE_BY_INDEX

Valid For

Extract and Replicat

Description

Use the `SET_COLUMN_VALUE_BY_INDEX` or `SET_COLUMN_VALUE_BY_NAME` function to modify a single column value without manipulating the entire data record. If the character session of the user exit is set with `SET_SESSION_CHARSET` to a value other than the default character set of the operating system, as defined in `ULIB_CS_DEFAULT` in the `ucharset.h` file, the character data that is exchanged between the user exit and the process is interpreted in the session character set.

A column value is set to the session character set only if the following is true:

- The column value is a SQL character type (`CHAR/VARCHAR2/CLOB`, `NCHAR/NVARCHAR2/NCLOB`), a SQL date/timestamp/interval/number type)
- The `column_value_mode` indicator is set to `EXIT_FN_CNVTED_SESS_CHAR_FORMAT`.

Syntax

```
#include "usrdecs.h"
short result_code;
column_def column;
ERCALLBACK (SET_COLUMN_VALUE_BY_INDEX, &column, &result_code);
```

Buffer

```
typedef struct
{
char *column_value;
unsigned short max_value_length;
unsigned short actual_value_length;
short null_value;
short remove_column;
short value_truncated;
short column_index;
char *column_name;
/* Version 3 CALLBACK_STRUCT_VERSION */
short column_value_mode;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION */
char requesting_before_after_ind;
```

```
char more_lob_data;
/* Version 3 CALLBACK_STRUCT_VERSION  */
ULibCharSet column_charset;
} column_def;
```

Input

`column_value`
A pointer to a buffer representing the new column value.

`actual_value_length`
The length of the new column value, in bytes. The actual length should not include the null terminator if the new column value is in ASCII format.

`null_value`
A flag (`0` or `1`) indicating whether the new column value is null. If the `null_value` flag is set to `1`, the column value in the data record is set to null.

`remove_column`
A flag (`0` or `1`) indicating whether to remove the column from a compressed update if it exists. A compressed update is one in which only the changed column values are logged, not all of the column values. This flag should only be set if the operation type for the record is `UPDATE_COMP_SQL_VAL` or `PK_UPDATE_SQL_VAL`.

`column_index`
The column index of the new column value to be copied into the data record buffer. Column indexes start at zero.

`column_value_mode`
Indicates the format of the column value.

`EXIT_FN_CHAR_FORMAT`
ASCII format: The value is a null-terminated ASCII (or EBCDIC) string (with a known exception for the sub-data type UTF16_BE, which is converted to UTF8.)

> **Note:**
>
> A column value might be truncated when presented to a user exit, because the value is interpreted as an ASCII string and is supposed to be null-terminated. The first value of `0` becomes the string terminator.

- Dates are in the format `CCYY-MM-DD HH:MI:SS.FFFFFF`, in which the fractional time is database-dependent.
- Numeric values are in their string format. For example, `123.45` is represented as `'123.45'`.
- Non-printable characters or binary values are converted to hexadecimal notation.
- Floating point types are output as null-terminated strings, to the first 14 significant digits.

EXIT_FN_RAW_FORMAT
Internal Oracle GoldenGate canonical format: This format includes a two-byte null indicator and a two-byte variable data length when applicable. No character-set conversion is performed by Oracle GoldenGate for this format for any character data type.

EXIT_FN_CNVTED_SESS_CHAR_FORMAT
User exit character set: This only applies if the column data type is:

- a character-based type, single or multi-byte
- a numeric type with a string representation

This format is not null-terminated.

source_or_target
One of the following indicating whether the source or target record is being modified.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

requesting_before_after_ind
Set when setting a column value on a record `io_type` of `UPDATE_COMP_PK_SQL_VAL` (primary key update). Use one of the following to indicate which portion of the primary key update is to be accessed. The default is `AFTER_IMAGE_VAL`.

- `BEFORE_IMAGE_VAL`
- `AFTER_IMAGE_VAL`

Output

None

Return Values

```
EXIT_FN_RET_BAD_COLUMN_DATA
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
EXIT_FN_RET_NOT_SUPPORTED
EXIT_FN_RET_INVALID_COLUMN_TYPE
```

4.48 SET_COLUMN_VALUE_BY_NAME

Valid For

Extract and Replicat

Description

Use the `SET_COLUMN_VALUE_BY_NAME` or `SET_COLUMN_VALUE_BY_INDEX` function to modify a single column value without manipulating the entire data record.

If the character session of the user exit is set with `SET_SESSION_CHARSET` to a value other than the default character set of the operating system, as defined in `ULIB_CS_DEFAULT` in the `ucharset.h` file, the character data that is exchanged between the user exit and the process is interpreted in the session character set.

A column value is set to the session character set only if the following is true:

- The column value is a SQL character type (CHAR/VARCHAR2/CLOB, NCHAR/NVARCHAR2/NCLOB), a SQL date/timestamp/interval/number type)
- The column_value_mode indicator is set to EXIT_FN_CNVTED_SESS_CHAR_FORMAT.

If the database is case-sensitive, object names must be specified in the same letter case as they are defined in the hosting database; otherwise, the case does not matter.

Syntax

```
#include "usrdecs.h"
short result_code;
column_def column;
ERCALLBACK (SET_COLUMN_VALUE_BY_NAME, &column, &result_code);
```

Buffer

```
typedef struct
{
char *column_value;
unsigned short max_value_length;
unsigned short actual_value_length;
short null_value;
short remove_column;
short value_truncated;
short column_index;
char *column_name;
/* Version 3 CALLBACK_STRUCT_VERSION */
short column_value_mode;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION */
char requesting_before_after_ind;
char more_lob_data;
/* Version 3 CALLBACK_STRUCT_VERSION  */
ULibCharSet column_charset;
} column_def;
```

Input

column_value
A pointer to a buffer representing the new column value.

actual_value_length
The length of the new column value, in bytes. The actual length should not include the null terminator if the new column value is in ASCII format.

null_value
A flag (0 or 1) indicating whether the new column value is null. If the null_value flag is set to 1, the column value in the data record is set to null.

remove_column
A flag (0 or 1) indicating whether to remove the column from a compressed update if it exists. A compressed update is one where only the changed column values are logged, not all of the column values. This flag should only be set if the operation type for the record is UPDATE_COMP_SQL_VAL or PK_UPDATE_SQL_VAL.

column_name
The name of the column that corresponds to the new column value to be copied into the data record buffer.

column_value_mode
Indicates the format of the column value.

EXIT_FN_CHAR_FORMAT
ASCII format: The value is a null-terminated ASCII (or EBCDIC) string (with a known exception for the sub-data type `UTF16_BE`, which is converted to UTF8.)

> **Note:**
>
> A column value might be truncated when presented to a user exit, because the value is interpreted as an ASCII string and is supposed to be null-terminated. The first value of `0` becomes the string terminator.

- Dates are in the format `CCYY-MM-DD HH:MI:SS.FFFFFF`, in which the fractional time is database-dependent.
- Numeric values are in their string format. For example, `123.45` is represented as `'123.45'`.
- Non-printable characters or binary values are converted to hexadecimal notation.
- Floating point types are output as null-terminated strings, to the first 14 significant digits.

EXIT_FN_RAW_FORMAT
Internal Oracle GoldenGate canonical format: This format includes a two-byte null indicator and a two-byte variable data length when applicable. No character-set conversion is performed by Oracle GoldenGate for this format for any character data type.

EXIT_FN_CNVTED_SESS_CHAR_FORMAT
User exit character set: This only applies if the column data type is:

- a character-based type, single or multi-byte
- a numeric type with a string representation

This format is not null-terminated.

source_or_target
One of the following indicating whether the source or the target data record is being modified.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

requesting_before_after_ind
Set when setting a column value on a record `io_type` of `UPDATE_COMP_PK_SQL_VAL` (primary key update). Use one of the following to indicate which portion of the primary key update is to be accessed. The default is `AFTER_IMAGE_VAL`.

- BEFORE_IMAGE_VAL
- AFTER_IMAGE_VAL

Output

None

Return Values

```
EXIT_FN_RET_BAD_COLUMN_DATA
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
EXIT_FN_RET_NOT_SUPPORTED
EXIT_FN_RET_INVALID_COLUMN_TYPE
```

4.49 SET_OPERATION_TYPE

Valid For

Extract and Replicat

Description

Use the SET_OPERATION_TYPE function to change the operation type associated with a data record. For example, a delete on a specified table can be turned into an insert into another table. The record header's before/after indicator is modified as appropriate for insert and delete operations.

Syntax

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (SET_OPERATION_TYPE, &record, &result_code);
```

Buffer

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

Input

`io_type`
Returned as one of the following for deletes, inserts, and updates, respectively:

```
DELETE_VAL
INSERT_VAL
UPDATE_VAL
```

For a compressed SQL update, the following is returned:

```
UPDATE_COMP_SQL_VAL
```

If the new operation type is an insert or delete, the before/after indicator for the record is set to one of the following:
Insert: `AFTER_IMAGE_VAL` (after image)
Delete: `BEFORE_IMAGE_VAL` (before image)

`source_or_target`
One of the following indicating whether to set the operation type for the source or target data record.

```
EXIT_FN_SOURCE_VAL
EXIT_FN_TARGET_VAL
```

Output

None

Return Values

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

4.50 SET_RECORD_BUFFER

Valid For

Extract and Replicat

Description

Use the `SET_RECORD_BUFFER` function for compatibility with user exits, and for complex data record manipulation. This function manipulates the entire record. It is best to modify individual column values, rather than the entire record, because the Oracle GoldenGate internal record formats must be known in order to accurately modify the data record buffer directly. To modify column values, use the `SET_COLUMN_VALUE_BY_INDEX` and `SET_COLUMN_VALUE_BY_NAME` functions. These functions are sufficient to handle most custom mapping within a user exit.

Syntax

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (SET_RECORD_BUFFER, &record_def, &result_code);
```

Buffer

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

Input

`buffer`
A pointer to the new record buffer. Typically, `buffer` is a pointer to a buffer of type `exit_rec_buf_def`. The `exit_rec_buf_def` buffer contains the actual record about to be processed by Extract or Replicat. The buffer is supplied when the call type is `EXIT_CALL_DISCARD_RECORD`. Exit routines can change the contents of this buffer, for example to perform custom mapping functions.
The content of the record buffer is not converted to or from the character set of the user exit. It is passed as-is.

`length`
The new length of the record buffer.

Output

None

Return Values

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
EXIT_FN_RET_NOT_SUPPORTED
```

4.52 SET_TABLE_NAME

Valid For

Extract and data pumps

Description

Use the `SET_TABLE_NAME` function to change the table name associated with a data record. For example, a delete on a specified table can be changed to an insert into a history table. You can change the table name only during Extract processing.

If the database is case-sensitive, object names must be specified in the same letter case as they are defined in the hosting database; otherwise, the case does not matter. Specify the full two-part or three-part table name.

Syntax

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (SET_TABLE_NAME, &record_def, &result_code);
```

Buffer

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

Input

`table_name`
A null-terminated string specifying the new table name to be associated with the data record.
If the character session of the user exit is set with `SET_SESSION_CHARSET` to a value other than the default character set of the operating system, as defined in `ULIB_CS_DEFAULT` in the `ucharset.h` file, the table name is interpreted in the session character set.

Output

None

Return Values

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

Part II

Oracle GoldenGate Microservices Architecture

You can use Oracle GoldenGate Microservices Architecture to configure and manage your data replication using an HTML user interface.

This part describes the commands, parameters, and options that you can use with the Oracle GoldenGate Microservices Architecture.

Topics:

- Oracle GoldenGate AdminClient Commands
  You can use the Oracle GoldenGate AdminClient commands to create data replications. This is the command line interface between you and Oracle GoldenGate functional components.

5
Oracle GoldenGate AdminClient Commands

You can use the Oracle GoldenGate AdminClient commands to create data replications. This is the command line interface between you and Oracle GoldenGate functional components.

Topics:

- Admin Client Commands
  These are the Admin Client commands that you can use including examples.

5.1 Admin Client Commands

These are the Admin Client commands that you can use including examples.

```
------------------------------------------------
!
Use the ! command to execute a previous Admin Client command without
modifications. To
display a list of previous commands, use the HISTORY command (see HISTORY).
The ! command without arguments executes the most recent command. Options enable
you to execute any previous command by specifying its line number substring.
Previous
commands can be executed again only if they were issued during the current
session of
Admin Client, because command history is not maintained from session to session.

n

Executes the command from the specified Admin Client line. Each Admin Client
command line is sequenced, beginning with 1 at the start of the session.

!

Examples

Example 1
! 9



------------------------------------------------
ADD AUTORESTART

Creates an automatic restart (autorestart) rule for Extract, Replicat, or both.

Syntax:

ADD AUTORESTART rule-name ( EXTRACT | REPLICAT | ER ) group-name-wildcard
  |[ RETRIES       ]
  |[ WAITSECONDS   ]
```

```
  |[ RESETSECONDS  ]

-----------------------------------------------
DELETE AUTORESTART
Removes an autorestart rule.

Syntax:
DELETE AUTORESTART rule-name


-----------------------------------------------
INFO AUTORESTART
Returns information about the specified autorestart rule.

Syntax:
INFO AUTORESTART rule-name-wildcard


-----------------------------------------------
ADD CHECKPOINTTABLE

Not valid for Replicat for Java, Oracle GoldenGate Applications Adapter, or
Oracle GoldenGate for Big Data.

Use ADD CHECKPOINTTABLE to create a checkpoint table in the target database.
Replicat uses the table to maintain a record of its read position in the trail
for recovery purposes.

The use of a checkpoint table is strongly recommended, because it causes
checkpoints to be part of the Replicat transaction. This allows Replicat to
recover more easily in certain circumstances than when a checkpoint file alone
is used. However, do not use a checkpoint table when configuring Replicat to
operate in integrated mode against an Oracle target database. It is not
required in that mode.

One table can serve as the default checkpoint table for all Replicat groups in
an Oracle GoldenGate instance if you specify it with the CHECKPOINTTABLE
parameter
in a GLOBALS file. More than one instance of Oracle GoldenGate (multiple
installations)
can use the same checkpoint table. Oracle GoldenGate keeps track of the
checkpoints
even when the same Replicat group name exists in different instances.

Use the DBLOGIN command to establish a database connection before using this
command.
Do not change the names or attributes of the columns in this table. You may,
however,
change table storage attributes.

For more information about using a checkpoint table, see Administering Oracle
GoldenGate.


table-name

The name of the checkpoint table to be created. The name cannot contain any
special characters, such as quotes, backslash, dollar sign, and percent symbol.

Record the name of the table, because you will need it to view statistics or
delete the
```

```
table if needed.

Example

ADD CHECKPOINTTABLE ggs.fin_check

-----------------------------------------------
ADD DISTPATH

Use ADD DISTPATH to create a distribution path.

path-name

The unique name of the distribution path you want to add.

source-uri

Specifies the source URI after SOURCE keyword to indicate where the data
is originated. The format of this URI contains the protocol
(currently only supports trail), hostname, port number of distribution server,
and location of the source trail files.

target-uri

Specifies the target URI after TARGET keyword to indicate where the data
will be sent to. The format of this URI contains the protocol (currently only
supports legacy ogg protocol, web socket ws protocol, and
secure web socket wss protocol), hostname, port number of receiver server,
and location of the target trail files.

TARGETTYPE

Specifies the target type in case the distribution path uses the legacy potocol.
This argument is only valid if the target URI schema is 'ogg://'.

Choose MANAGER if the target is a legacy deployment with a manager running.

Choose COLLECTOR if the target is a legacy deployment with a static collector
running.

Choose RECVSRVR if the target is an Microservices Architecture deployment with a
Receiver Server running.

Examples

Example 1

ADD DISTPATH path1  SOURCE trail://localhost:9002/services/v2/sources?trail=a1
                    TARGET wss://localhost:9003/services/v2/targets?trail=t1

Example 2

The target trail must specify the directory that contains your trail files. The
default
dirdat directory is used in this example.

ADD DISTPATH path1  SOURCE trail://localhost:9002/services/v2/sources?trail=a1
                    TARGET ogg://localhost:9003/services/v2/targets?
trail=dirdat/t1
                    TARGETTYPE MANAGER
```

```
Example 3

A fully-qualified alias.

ws://domain+alias@host:port

Example 4

An alias from a default domain.
ws://alias@host:port

Example 5

A credential store is not used; DS base64-encode user:password into the
Authorization header.

ws://user:passwd@host:port


------------------------------------------------
ADD EXTRACT

Use ADD EXTRACT to create an Extract group. Oracle GoldenGate supports up to
5,000
concurrent Extract and Replicat groups per instance of Oracle GoldenGate. At the
supported level, all groups can be controlled and viewed in full with commands
such
as the INFO and STATUS commands. Oracle GoldenGate recommends keeping the
combined
number of Extract and Replicat groups at the default level of 300 or below in
order
to manage your environment effectively.

group-name
The name of the Extract group. The name of an Extract group can contain
up to eight characters.

SOURCEISTABLE

Creates an Extract task that extracts entire records from the database
for an initial load using the Oracle GoldenGate direct load method or
the direct bulk load to SQL*Loader method. If SOURCEISTABLE is not
specified, ADD EXTRACT creates an online change-synchronization process,
and one of the other data source options must be specified. When using
SOURCEISTABLE, do not specify any service options. Task parameters
must be specified in the parameter file.

For more information about initial load methods, see Administering Oracle
GoldenGate.

TRANLOG

Specifies the transaction log as the data source; for classic capture only.

INTEGRATED TRANLOG

Adds this Extract in integrated capture mode. In this mode, Extract
integrates with the database logmining server, which passes logical change
records (LCRs) directly to Extract. Extract does not read the redo log. After
using INTEGRATED TRANLOG, use the REGISTER EXTRACT command. For information
about integrated capture, see Using Oracle GoldenGate
```

```
for Oracle Database.

BEGIN {NOW | begin-datetime}
Specifies a timestamp in the data source at which to begin processing.

    NOW

    NOW specifies the time at which the ADD EXTRACT command is issued.

    YYYY-MM-DDThh:mm:ssZ

    A date and time (timestamp) in the given form. For example,
2017-07-14T14:54:45Z.

EXTSEQNO sequence-number

Valid for a primary Extract in classic capture mode for Oracle. Not supported
for an Oracle Extract in integrated mode. Specifies the sequence number of an
Oracle redo log at which to begin capturing data. Contact Oracle Support before
using this option.

EXTRBA offset-number

Specifies the relative byte address within a transaction
log at which to begin capturing data.

SCN

Starts Extract at the transaction in the redo log
that has the specified Oracle system change number (SCN). This option
is valid for Extract both in classic capture and integrated modes.

THREADS threads-number

Valid for classic capture mode. Specifies the number of producer
threads that Extract maintains to read redo logs.
Required in an Oracle RAC configuration to specify the number of producer
threads. These are the Extract threads that read the different redo logs
on the various RAC nodes. The value must be the same as the number of nodes
from which you want to capture redo data.

DESC description

Specifies a description of the group, such as 'Extracts account_tab on
Serv1'. Enclose the description within single quotes.

Examples

Example 1

The following creates an integrated capture Extract group.
ADD EXTRACT finance, INTEGRATED TRANLOG, BEGIN NOW

Example 2

The following creates an initial-load Extract named load.

ADD EXTRACT load, SOURCEISTABLE

Example 3
```

The following examples create and position Extract at a specific Oracle system change number (SCN) in the redo log.

```
ADD EXTRACT finance INTEGRATED TRANLOG SCN 123456
```

ADD EXTTRAIL

Use ADD EXTTRAIL to create a trail for online processing on the local system and:

* Associate it with an Extract group.

* Assign a maximum file size.

trail-name

The relative or fully qualified path name of the trail. The trail name can contain only two characters. Oracle GoldenGate appends this name with a nine-digit sequence number whenever a new file is created. For example, a trail named var/lib/data/tr would have files named var/lib/data/tr000000001, var/lib/data/tr000000002, and so forth.

group-name

The name of the Extract group to which the trail is bound. Only one Extract process can write data to a trail.

MEGABYTES megabytes-number

The maximum size, in megabytes, of a file in the trail. The default is 500.

SEQNO sequence-number

Specifies that the first file in the trail will start with the specified trail sequence number. Do not include any zero padding. For example, to start at sequence 3 of a trail named tr, specify SEQNO 3. The actual file would be named /ggs/var/lib/data/tr000000003. This option can be used during troubleshooting when Replicat needs to be repositioned to a certain trail sequence number. It eliminates the need to alter Replicat to read the required sequence number.

Example

```
ADD EXTTRAIL /ggs/var/lib/data/aa, EXTRACT finance, MEGABYTES 200
```

ADD HEARTBEATTABLE

Use ADD HEARTBEATTABLE to create the objects necessary to use the automatic heartbeat functionality. This command:

* creates a heartbeat seed table, heartbeat table, and heartbeat history table,

* creates the GG_LAG and GG_LAG_HISTORY views,

* creates the GG_UPDATE_HB_TAB and GG_PURGE_HB_TAB procedures that are called by

```
the scheduler jobs,

* creates the scheduler jobs that periodically update the heartbeat and seed
table,
and purge the history table,

* populates the seed table.


The default seed, heartbeat, and history table names are GG_HEARTBEAT_SEED,
GG_HEARTBEAT,
and GG_HEARTBEAT_HISTORY respectively. The tables, procedures, and scheduler
jobs are
created in the GGSCHEMA mentioned in GLOBALS file. The default names can be
overridden
by specifying HEARTBEATTABLE hbschemaname.hbtablename in the GLOBALS file. In
this
case, the tables, procedures, and jobs are created in the schema, hbschemaname.
The
seed and history table are created by appending a _SEED and _HISTORY to the
table,
hbtablename.

This command requires a DBLOGIN. On a CDB database, a PDB login is required.

The ADD HEARTBEATTABLE has to be performed in every PDB that you want
to generate heartbeats for in CDB mode.


FREQUENCY frequency-seconds-number

Specifies how often the heartbeat seed table and heartbeat table are updated.
For example, how frequently heartbeat records are generated. The default is
60 seconds.

RETENTION_TIME retention-days-number

Specifies when heartbeat entries older than the retention time in the history
table are purged. The default is 30 days.

PURGE_FREQUENCY purge-frequency-days-number

Specifies how often the purge scheduler is run to delete table entries that
are older than the retention time from the heartbeat history. The default
is 1 day.

PARTITIONED

Enables partitioning on the heartbeat history table. The column for the
heartbeat
time stamp received is used to partition the table with an interval of one day.
By default the heartbeat history table is not partitioned.

NOADDTRANDATA

Disables supplemental logging for the heartbeat table and the heartbeat seed
table.
By default supplemental logging is enabled for both tables.

TARGETONLY
```

Disables supplemental logging on both heartbeat seed and heartbeat tables and it does not create scheduler job for updating heartbeat table.

Examples

Example 1

The following command creates default heartbeat tables, procedures, and jobs.

ADD HEARTBEATTABLE

Example 2

The following command creates the heartbeat tables, procedures, and jobs with custom frequency, retention time, and purge frequency.

ADD HEARTBEATTABLE, frequency 120, retention_time 10, purge_frequency 2

Example 3

ADD HEARTBEATTABLE PARTITIONED

It will setup all the heartbeat tables and the heartbeat history table will be partitioned.

Example 4

ADD HEARTBEATTABLE NOADDTRANDATA

It will setup all the heartbeat tables and supplemental logging will be disabled for the heartbeat table and heartbeat seed table.

ADD MASTERKEY

Use the ADD MASTERKEY command to add a master key to a master-key wallet. The master key is used by Extract and Replicat to encrypt the encryption keys that secure data being sent across the network and in the trail files, so that those keys can be sent to, and used, by downstream processes. The master key omits the need to use wallet storage for the keys that actually encrypt the data.

The wallet remains open throughout the same session in which the command was issued.

The successful completion of this command returns a message similar to the following:

2017-07-11T12:40:03Z ERROR OGG-06137 Master key 'OGG_DEFAULT_MASTERKEY' does not exist in Oracle Wallet: '/u02/ogg/Local/var/lib/wallet/cwallet.sso'.
2017-07-11T12:40:03Z INFO OGG-06142 Created version 1 of master key 'OGG_DEFAULT_MASTERKEY' in Oracle Wallet '/u02/ogg/Local/var/lib/wallet/cwallet.sso'.

OGG (http://localhost:9100 Local) 6 add masterkey
2017-07-11T12:41:20Z INFO OGG-06143 Master key 'OGG_DEFAULT_MASTERKEY' already exists in Oracle Wallet 'Local'.

After adding a master key to a wallet that is not maintained centrally on

shared storage, the updated wallet must be copied to all of the other
systems in the Oracle GoldenGate configuration that use this wallet.
Before doing so, Extract must be stopped and then all of the downstream
Oracle GoldenGate processes must be allowed to finish processing their
trails and then be stopped. After the wallet is copied into place, the
processes can be started again. For detailed instructions, see Administering
Oracle GoldenGate.

Example

ADD MASTERKEY

ADD PROCEDURETRANDATA

Enables procedure-level supplemental logging on Oracle database.

Example

ADD PROCEDURETRANDATA

ADD REPLICAT

Use ADD REPLICAT to create a Replicat group. Unless SPECIALRUN is specified,
ADD REPLICAT creates an online process group that creates checkpoints so
that processing continuity is maintained from run to run.

Oracle GoldenGate supports up to 5,000 concurrent Extract and Replicat groups
per instance of Oracle GoldenGate. At the supported level, all groups
can be controlled and viewed in full with commands such as the INFO and
STATUS commands. Oracle GoldenGate recommends keeping the number of Extract and
Replicat groups (combined) at the default level of 300 or below in order to
manage your environment effectively.
Unless the INTEGRATED
option is used, this command creates a Replicat group in non-integrated mode.

group_name

The name of the Replicat group. The name of a coordinated or parallel Replicat
group
can contain a maximum of five characters. The name of a regular Replicat
group can contain up to eight characters.

INTEGRATED

Creates the Replicat in integrated mode. Without this option,
ADD REPLICAT creates the Replicat in non-integrated (classic) mode. In this
mode, the Replicat process leverages the apply processing functionality that
is available within the Oracle database. In this mode, Replicat operates
as follows:

* Reads the Oracle GoldenGate trail.

* Performs data filtering, mapping, and conversion.

* Constructs logical change records (LCR) that represent source database DML
or DDL transactions (in committed order).

* Attaches to a background process in the target database known as a database inbound server by means of a lightweight streaming interface.

* Transmits the LCRs to the inbound server, which applies the data to the target database.

Do not use INTEGRATED with the SPECIALRUN or EXTFILE options. INTEGRATED must be used for an online change-synchronization Replicat that reads from a local EXTTRAIL-specified trail.
Integrated Replicat does not require a checkpoint table (ADD CHECKPOINTTABLE command and CHECKPOINTTABLE parameter). When in integrated mode, Replicat does n ot support the following parameters:

* BULKLOAD(Do not use integrated Replicat as an initial-load Replicat.)

* SPECIALRUN

* GENLOADFILES

* SHOWSYNTAX

* MAXTRANSOPS(is ignored)

PARALLEL

Adds the Replicat in parallel mode. In this mode, Replicat applies transactions in
parallel to improve the performance. It takes into account dependencies between transactions. PARALLEL INTEGRATED adds the parallel Replicat in integrated mode, which like Integrated Replicat leverages the apply processing functionality that is
available within the Oracle Database. You must use a checkpointtable with this Replicat.

COORDINATED [maxthreads-number]

Creates the Replicat in coordinated mode. A coordinated Replicat is multithreaded to enable parallel processing. This option adds the coordinator (identified by the group name itself) and the maximum number of processing threads that are specified by default or with MAXTHREADS. Dependencies are computed and coordinated by the coordinator, and the SQL processing is performed by the threads.

To create a COORDINATED Replicat, a checkpoint table is required.

Do not use COORDINATED with the SPECIALRUN or EXTFILE options. COORDINATED must be used for an online change-synchronization Replicat that reads from a local EXTTRAIL-specified trail. For more information about coordinated Replicat, see Administering Oracle GoldenGate.

Note:
The group name of a coordinated Replicat can contain only five characters.

maxthreads-number

Specifies the maximum number of processing threads that this Replicat group can spawn. These threads are all created on startup, but depending on what is specified in the MAP statements in the parameter file, some or

```
    all of these threads will process the workload at any given time. As a
    general rule, specify twice the number of threads that you specify in the
    MAP statements when you partition the workload. This allows you to add
    threads in the event that the workload increases, without having to drop
    and recreate the Replicat group.

    The default number of threads is 25 if maxthreads is omitted. The maximum
    number of threads is 500.

    The maxthreads option has a relationship to the MAXGROUPS parameter.
MAXGROUPS controls
    the maximum number of process groups (Extract and Replicat) allowed per
    instance of Oracle GoldenGate. Each Replicat thread is considered a Replicat
    group in the context of MAXGROUPS. Therefore, the number of Extract and
    Replicat groups in the Oracle GoldenGate instance, plus the value of
    maxthreads, cannot exceed the value of MAXGROUPS.

SPECIALRUN

Creates a Replicat special run as a task. Either SPECIALRUN, EXTFILE, or
EXTTRAIL is required. When Extract is in SPECIALRUN mode, do not start
Replicat with the START REPLICAT command. Do not use this option
with the INTEGRATED or COORDINATED option.

EXTFILE file-name

Specifies the relative or fully qualified name of an Extract file that is
specified with RMTFILE in the Extract parameter file. Do not use this option
with the INTEGRATED option.

EXTTRAIL trail-name

Specifies the relative or fully qualified name of a trail that was created
with the ADD RMTTRAIL or ADD EXTTRAIL command.

BEGIN {NOW | begin-datetime }

Defines an initial checkpoint in the trail.

    NOW
    Begins replicating changes from the time when the group is created.

    begin-datetime
    Begins extracting changes from a specific time.

EXTSEQNO sequence-number

Specifies the sequence number of the file in a trail in which to begin
processing data. Specify the sequence number, but not any zeroes used for
padding. For example, if the trail file is ggs/var/lib/data/aa000026, you
would specify EXTSEQNO 26. If not specified, the default value is zero.
By default, processing begins at the beginning of a trail unless this option
is used. Contact Oracle Support before using this option.

EXTRBA offset-number

Specifies the relative byte address within the trail file that is specified
by EXTSEQNO. Contact Oracle Support before using this option.

CHECKPOINTTABLE table-name
```

```
Not valid for Oracle GoldenGate Applications Adapter or Oracle GoldenGate for
Big Data.

Specifies that this Replicat group will write checkpoints to the specified
table in the database. Include the owner and table name, as in hr.hr_checkpoint.
This argument overrides any default CHECKPOINTTABLE specification in the GLOBALS
file. The table must first be added with the ADD CHECKPOINTTABLE command. Do not
use this option with the INTEGRATED option.

DESC description

Specifies a description of the group, such as 'Loads account_tab on Serv2'.
Enclose the description within quotes.

Examples

Example 1

ADD REPLICAT sales, EXTTRAIL var/lib/data/rt

Example 2

ADD REPLICAT sales, INTEGRATED, EXTTRAIL var/lib/data/rt

Example 3

This example creates Replicat in coordinated mode. It indicates that up
to 100 threads can be employed in parallel at any given point in processing.

ADD REPLICAT sales, COORDINATED MAXTHREADS 100, EXTTRAIL var/lib/data/rt

------------------------------------------------
ADD RMTTRAIL

Use ADD RMTTRAIL to create a trail for online processing on a remote system and:

* Assign a maximum file size.

* Associate the trail with an Extract group.

In the parameter file, specify a RMTHOST entry before any RMTTRAIL entries
to identify the remote system.

    Note:
    The RMTTRAIL size (Target Trail) must be greater than or equal to the
    EXTTRAIL size (Source Trail), due to trail encryption requirements.

trail-name

The relative or fully qualified path name of the trail. The actual trail
name can contain only two characters. Oracle GoldenGate appends this name
with a nine-digit sequence number whenever a new file is created. For example,
a trail named ./ var/lib/data/trwould have files named ./var/lib/data/
tr000000001,
./var/lib/data/ tr000000002, and so forth.

group-name

The name of the Extract group to which the trail is bound. Only one primary
Extract process can write data to a remote trail.
```

```
MEGABYTES megabytes-number

The maximum size, in megabytes, of a file in the trail. The default is 500.

SEQNO sequence-number

Specifies that the first file in the trail will start with the specified
trail sequence number. Do not include any zero padding. For example, to start
at sequence 3 of a trail named tr, specify SEQNO 3. The actual file would
be named /ggs/var/lib/data/ tr000000003. This option can be used during
troubleshooting when Replicat needs to be repositioned to a certain trail
number. It eliminates the need to alter Replicat
 to read the required
sequence number.

Example

ADD RMTTRAIL var/lib/data/aa, EXTRACT finance, MEGABYTES 200


-----------------------------------------------
ADD SCHEMATRANDATA

Use ADD SCHEMATRANDATA to enable schema-level supplemental logging for a schema.
ADD SCHEMATRANDATA acts on all of the current and future tables in a given
schema to automatically log a superset of available keys that Oracle GoldenGate
needs for row identification.

ADD SCHEMATRANDATA does the following:

* Enables Oracle supplemental logging for new tables created with a
CREATE TABLE.

* Updates supplemental logging for tables affected by an ALTER TABLE
to add or drop columns.

* Updates supplemental logging for tables that are renamed.

* Updates supplemental logging for tables for which unique or primary keys
are added or dropped.


By default, ADD SCHEMATRANDATA logs the key columns of a table in the
following order of priority:

1. Primary key

2. In the absence of a primary key, all of the unique keys of the table,
including those that are disabled, unusable or invisible. Unique keys that
contain ADT member columns are also logged. Only unique keys on virtual
columns (function-based indexes) are not logged.

3. If none of the preceding exists, all scalar columns of the table are
logged. (System generated row-OIDs are always logged.)


ADD SCHEMATRANDATA also supports the conditional or unconditional logging
requirements for using integrated Replicat.

Use ADD SCHEMATRANDATA in the following cases:
```

```
* For all tables that are part of an Extract group that is to be
configured for integrated capture. ADD SCHEMATRANDATA ensures that the
correct key is logged by logging all of the keys.

* For all source tables that will be processed in an integrated Replicat
group. Options are provided that enable the logging of the primary, unique,
and foreign keys to support the computation of dependencies among relational
tables being processed through different apply servers.

* When DDL replication is active and DML is concurrent with DDL that creates
new tables or alters key columns. It best handles scenarios where DML can be
applied to objects very shortly after DDL is issued on them. ADD SCHEMATRANDATA
causes the appropriate key values to be logged in the redo log atomically with
each DDL operation, thus ensuring metadata continuity for the DML when it is
captured from the log, despite any lag in Extract processing.

Database-level Logging Requirements for Using ADD SCHEMATRANDATA
Oracle strongly encourages putting the source database into forced logging
mode and enabling minimal supplemental logging at the database level when
using Oracle GoldenGate. This adds row chaining information, if any exists,
to the redo log for update operations.

Additional Considerations for Using ADD SCHEMATRANDATA

* Before using ADD SCHEMATRANDATA, issue the DBLOGIN command. The user who
issues the command must be granted the Oracle Streams administrator privilege.

SQL exec dbms_streams_auth.grant_admin_privilege('user')

* ADD SCHEMATRANDATA can be used instead of the ADD TRANDATA command when DDL
replication is not enabled. Note, however, that if a table has no primary key
but has multiple unique keys, ADD SCHEMATRANDATA causes the database to log all
of the unique keys. In such cases, ADD SCHEMATRANDATA causes the database to log
more redo data than does ADD TRANDATA. To avoid the extra logging, designate one
of the unique keys as a primary key, if possible.

* For tables with a primary key, with a single unique key, or without a key, ADD
 SCHEMATRANDATA adds no additional logging overhead, as compared to ADD
TRANDATA.

* If you must log additional, non-key columns of a specific table (or tables)
for
use by Oracle GoldenGate, such as those needed for FILTER statements and KEYCOLS
clauses in the TABLE and MAP parameters, issue an ADD TRANDATA command for those
columns. That command has a COLS option to issue table-level supplemental
logging
for the columns, and it can be used in conjunction with ADD SCHEMATRANDATA.

schema-name

The schema for which you want the supplementary key information to be logged.
Do not use a wildcard. To issue ADD SCHEMATRANDATA for schemas in more than one
pluggable database of a multitenant container database, log in to each pluggable
database separately with DBLOGINand then issue ADD SCHEMATRANDATA.

NOSCHEDULINGCOLS | ALLCOLS

These options control supplemental logging for an Oracle target database. You
can
use these options together though the latter option is used. For example, with
the
```

```
ADD SCHEMATRANDATA oggadm_ext ALLCOL NOSCHEDULINGCOLS command the
NOSCHEDULINGCOLS
option would be used.

    NOSCHEDULINGCOLS

    Disables the logging of scheduling columns. By default, ADD SCHEMATRANDATA
    enables the unconditional logging of the primary key and the conditional
    supplemental logging of all unique key(s) and foreign key(s) of all current
    and future tables in the given schema. Unconditional logging forces the
primary
    key values to the log whether or not the key was changed in the current
    operation. Conditional logging logs all of the column values of a foreign or
    unique key if at least one of them was changed in the current operation. The
    integrated Replicat primary key, unique keys, and foreign keys must all be
    available to the inbound server to compute dependencies. For more
information
    about integrated Replicat, see Using Oracle GoldenGate for
    Oracle database.

    ALLCOLS

    Enables the unconditional supplemental logging of all supported key and non-
key
    columns for all current and future tables in the given schema. This option
    enables the logging of the keys required to compute dependencies, plus
columns
    that are required for filtering, conflict resolution, or other purposes.
Columns
    like LOB, LONG, and ADT are not included.

ALLOWNONVALIDATEDKEYS

It includes NON VALIDATED and NOT VALID primary keys in the supplemental
logging.
These keys override the normal key selection criteria that is used by Oracle
GoldenGate. If the GLOBALS parameter ALLOWNONVALIDATEDKEYS is being used, ADD
SCHEMATRANDATA runs with ALLOWNONVALIDATEDKEYS whether or not it is specified.
By default, NON VALIDATED and NOT VALID primary keys are not logged.

PREPARECSN {WAIT | LOCK | NOWAIT | NONE}

Valid for Oracle for both DML and DDL. Automatically prepares the tables at the
source
so the Oracle Datapump Export dump file will includes Instantiation CSNs.
Replicat uses
the per table instantiation CSN set by the Oracle Datapump (on import) to filter
out
trail records.

    WAIT

    Wait for any in-flight transactions and prepare table instantiation.

    LOCK

    Put a lock on the table (to prepare for table instantiation).

    NOWAIT

    Default behavior, preparing for instantiation is done immediately.
```

```
    NONE

    No instantiation preparation occurs.

Example

The following enables supplemental logging for the schema scott.

ADD SCHEMATRANDATA scott


------------------------------------------------
ADD TRACETABLE

Use ADD TRACETABLE to create a trace table in the Oracle database. The trace
table
must reside in the schema of the Oracle GoldenGate Extract user, as configured
with
the USERID or USERIDALIAS parameter. The trace table prevents Replicat
transactions
from being extracted again in a bidirectional synchronization configuration.

Use the DBLOGIN command to establish a database connection before using this
command.

The trace table has the following description.

Description of trace table

Name          Null?       Type            Description
GROUP_ID      NOT NULL VARCHAR2(8)  The name of the Replicat group or special
run process.
DB_USER               VARCHAR2(30) The user ID of the Replicat group or special
run process.
LAST_UPDATE           DATE            The timestamp of the transaction.

table-name

Use to specify a trace table with a schema name that is different from the
default of GGS_TRACE. The owner must be the same owner that is specified with
the
USERID or USERIDALIAS parameter in the Extract parameter file.
To use the default name, omit this argument. Whenever possible, use the default
table
name. When using a trace table name other than the default of GGS_TRACE, specify
it
with the TRACETABLE parameter in the Extract and Replicat parameter files.
Record the
name, because you will need it for the parameter files and to view statistics or
delete the table.

Example

ADD TRACETABLE finance.ora_trace


------------------------------------------------
ADD TRANDATA

Use ADD TRANDATA to enable Oracle GoldenGate to acquire the transaction
```

```
information
that it needs from the transaction records.

Before using this command, use the DBLOGIN command to establish a database
connection.

For other supported databases, this functionality may exist already or must be
configured through the database interface.

By default, ADD TRANDATA enables the unconditional logging of the primary
key and the conditional supplemental logging of all unique key(s) and foreign
key(s)
of the specified table.

If possible, use the ADD SCHEMATRANDATA command rather than the ADD TRANDATA
command.
The ADD SCHEMATRANDATA command ensures replication continuity should DML occur
on an
object for which DDL has just been performed. You can exclude objects from the
schema specification by using the exclusion parameters.

To use the Oracle GoldenGate DDL replication feature, you must use the ADD
SCHEMATRANDATA command to log the required supplemental data.

When using ADD SCHEMATRANDATA, you can use ADD TRANDATA with the COLS option to
any non-key columns, such as those needed for FILTER statements and KEYCOLS
clauses
in the TABLEand MAP parameters.

    Note:
    It is possible to use ADD TRANDATA when DDL support is enabled, but
    only if you can stop DML on all tables before DDL is performed on them or,
if
    that is not possible, you can guarantee that no users or applications will
issue
    DDL that adds new tables whose names satisfy an object specification in a
TABLE
    or MAP statement. There must be no possibility that users or applications
will
    issue DDL that changes the key definitions of any tables that are already in
    the Oracle GoldenGate configuration.

Oracle recommends putting the source database into forced logging mode
and enabling minimal supplemental logging at the database level when using
Oracle
GoldenGate. This adds row chaining information, if any exists, to the redo log
for
update operations.

Note the following:

* If any of the logging details change after Oracle GoldenGate starts extracting
data, you must stop and then start the Extract process that is reading from the
affected table before any data is changed.

* When creating a supplemental log group with ADD TRANDATA, Oracle GoldenGate
appends the object ID to a prefix of GGS_, for example GGS_18342.

Use he two-part or three-part name specification for all supported databases
except an
Oracle multitenant container database. A wildcard can be used for any component.
```

Used with a wildcard, ADD TRANDATA filters out names that match the names of system
objects. To use ADD TRANDATA for objects that are not system objects but have names
that match those of system objects in a wildcard pattern, issue ADD TRANDATA for those objects without using a wildcard.

NOSCHEDULINGCOLS | ALLCOLS

These options satisfy the logging requirements of an integrated Replicat that will
be processing the tables that you are specifying with ADD TRANDATA.

NOSCHEDULINGCOLS

Disables the logging of scheduling columns. By default, ADD TRANDATA enables the unconditional logging of the primary key and the conditional supplemental logging
of all unique key(s) and foreign key(s) of the specified table. Unconditional logging
forces the primary key values to the log whether or not the key was changed in the
current operation. Conditional logging logs all of the column values of a foreign or
unique key if at least one of them was changed in the current operation. The primary
key, unique keys, and foreign keys must all be available to the inbound server to
compute dependencies.

ALLCOLS

Enables the unconditional supplemental logging of all of the key and non-key columns of the table. This option enables the logging of the keys required to compute dependencies, plus all other columns for use in filtering, conflict resolution, or other purposes.

ALLOWNONVALIDATEDKEYS

It includes NON VALIDATED and NOT VALID primary keys in the supplemental logging.
These keys override the normal key selection criteria that is used by Oracle GoldenGate. If the GLOBALS parameter ALLOWNONVALIDATEDKEYS is being used, ADD SCHEMATRANDATA runs wit ALLOWNONVALIDATEDKEYS whether or not it is specified.
By default, NON VALIDATED and NOT VALID primary keys are not logged.

PREPARECSN {WAIT | LOCK | NOWAIT | NONE}

Automatically prepares the tables at the source so the Oracle Datapump Export dump file will includes Instantiation CSNs. Replicat uses the per table instantiation CSN set by the Oracle Datapump (on import) to filter out trail records.

WAIT

Wait for any in-flight transactions and prepare table instantiation.

LOCK

Put a lock on the table (to prepare for table instantiation).

```
    NOWAIT

    Default behavior, preparing for instantiation is done immediately.

    NONE

    No instantiation preparation occurs.

Examples

Example 1

The following example causes the primary key to be logged for an Oracle table.

ADD TRANDATA finance.acct

Example 2

The following example enables the unconditional supplemental logging of all of
the
key and non-key columns for the table named acct.

ADD TRANDATA finance.acct ALLCOLS

Example 3

The following example adds logging though does not prepare the table for
instantiation.

ADD TRANDATA finance.acct PREPARECSN NONE


------------------------------------------------
ALLOWNESTED

Use the ALLOWNESTED command to enable the use of nested OBEY files. A nested
OBEY file is one that contains another OBEY file. There is no maximum of the
number of nested files.

The default behavior in Admin Client is to disallow nested OBEY files.
ADD CREDENTIALSTORE


------------------------------------------------
Use the ADD CREDENTIALSTORE command to create a credential store. The
credential store manages user IDs and their encrypted passwords (together
known as credentials) that are used by Oracle GoldenGate processes to
interact with the local database. The credential store eliminates the
need to specify user names and clear-text passwords in the Oracle GoldenGate
parameter files. An optional alias can be used in the parameter file
instead of the user ID to map to a userid-password pair in the credential
store.

The credential store is implemented as an autologin wallet within the
Oracle Credential Store Framework (CSF). The use of an LDAP directory
is not supported for the Oracle GoldenGate credential store. The autologin
wallet supports automated restarts of Oracle GoldenGate processes without
requiring human intervention to supply the necessary passwords.

ADD CREDENTIALSTORE creates an empty credentials store in the location that
```

is specified with the CREDENTIALSTORELOCATION parameter in the GLOBALS file, if used, or otherwise in the default location of dircrdin the Oracle GoldenGate installation directory. A credential store can be shared by multiple instances (installations) of Oracle GoldenGate on the same or different systems. Store a shared credential store in a shared file system, and specify this location in each Oracle GoldenGate instance by using the CREDENTIALSTORELOCATION parameter in each GLOBALS parameter file.

Only one credential store can be used at a time by any given instance of Oracle GoldenGate. For example, you can have a credential store named /home/ogg/ credentialsand a credential store named /test/ogg/credentials, but only one can be used at runtime by a given instance of Oracle GoldenGate. You can stop the processes to switch to a different credential store, but make certain to update the CREDENTIALSTORELOCATION parameter in each GLOBALS parameter file, and change the USERIDALIAS parameters to specify different aliases if needed.

The use of a credential store is not supported for the iSeries, z/OS, and NonStop platforms.

For more information about Oracle GoldenGate security options, see Administering Oracle GoldenGate.

Syntax

ADD CREDENTIALSTORE

ALTER CREDENTIALSTORE

Use the ALTER CREDENTIALSTORE command to manage user ID and password pairs in the credential store. This command enables you to add credentials to the credential store and to specify different aliases for a user. For more information about Oracle GoldenGate security options, see Administering Oracle GoldenGate.

ADD USER userid

Adds the specified user and its alias to the credential store. If the ALIAS option is not used, the alias defaults to the user name. A credential can only be entered once unless the ALIAS option is used to specify a different alias for each one. Unless the PASSWORD option is used, the command prompts for the password of the specified user. The user can be an actual user name or a SQL*Net connect string.

REPLACE USER userid

Changes the password of the specified user. If the ALIAS option is not used, the alias defaults to the user name. You cannot change the alias or domain of a user with this option, but you can use the ADD USER option to add a new entry for the user under the desired ALIAS or DOMAIN. Unless the PASSWORD option is used, the command prompts for the new password for the specified user.

DELETE USER userid

Removes the credential for the specified user from the credential store. If the ALIAS option is not used, the alias defaults to the user name.

PASSWORD password

```
The user's password. The password is echoed (not hidden) when this
option is used. If this option is omitted, the command prompts for the
password, which is hidden as it is typed (recommended as more secure).

ALTER CREDENTIALSTORE ADD USER scott
Password: ********

ALIAS alias

Specifies an alias for the user name. Use this option if you do not want
the user name to be in a parameter file or command. If ALIAS is not used,
the alias defaults to the USER name, which then must be used in parameter
files and commands where a login is required. You can create multiple
entries for a user, each with a different alias, by using the ADD USER option
with ALIAS.

DOMAIN domain

Enables the same alias to be used by multiple Oracle GoldenGate installations
that use the same credential store. The default domain is Oracle GoldenGate.
For example, the administrators of system 1 might not want system 2 to have
access to the same credentials that are used on system 1. Those credentials
can be stored as ALIAS extract, for example, under DOMAIN system1, while a
different set of credentials can be stored for ALIAS extractunder DOMAIN
system2.

Examples

Example 1

This example adds a user named scott but omits the PASSWORD specification,
so the command prompts for Scott's password.

ALTER CREDENTIALSTORE ADD USER scott
Password: ********

Example 2

This example adds the user scott with his password tiger and specifies an alias
for scott that is named scsm2.

ALTER CREDENTIALSTORE ADD USER scott PASSWORD tiger ALIAS scsm2

Example 3

This example adds the user scott under the domain of support.

ALTER CREDENTIALSTORE ADD USER scott ALIAS scsm3 DOMAIN support
Password: ********

Example 4

This example issues two ALTER CREDENTIALSTORE commands, each of which
adds a scott entry, but with a different alias.

ALTER CREDENTIALSTORE ADD USER scott ALIAS scsm2
Password: ********
ALTER CREDENTIALSTORE ADD USER scott ALIAS scsm3
Password: ********

Example 5
```

```
The following shows how the DELETE USER option works with and without the
ALIAS option.

The following command deletes the user1 entry for which the ALIAS is the
same as the user name.

ALTER CREDENTIALSTORE DELETE USER user1

The following command deletes the entry for user user1 that is associated
with the alias alias1.

ALTER CREDENTIALSTORE DELETE USER user1 ALIAS alias1

Example 6

This example uses a SQL*Net connect string as the user value. In this case,
the PASSWORD option is omitted. The person issuing the command will be
prompted for the password, which is hidden.
ALTER CREDENTIALSTORE ADD USER oggext1@ora1 ALIAS ora1


-------------------------------------------------
ALTER DISTPATH

Changes the attributes of a distribution path.

path-name

The name of the distribution path.

BEGIN {NOW | SEQNO sequence-number RBA relative-byte-address | begin-datetime}
Specifies a timestamp in the data source at which to begin processing.

    NOW

    NOW specifies the time at which the ADD EXTRACT command is issued.

    sequence-number relative-byte-address

    The sequence number of an Oracle redo log and RBA within that log at which
to begin
    capturing data.

    begin-datetime

    A date and time (timestamp) in the given form. For an Extract in integrated
mode, the
    timestamp value must be greater than the timestamp at which the Extract was
registered
    with the database.

TARGET options

At least one TARGET option must be specified.

CRITICAL option

Indicates that the distribution path is critical to the deployment. The default
is false.
```

```
FILTER options

At least one FILTER option must be specified.
By default the RELATION between the filters is OR and the action is KEEP.

     CHUNKIDS chunk-ids

     Specify a rule to filter records by their chunk ID (sharding).
     The list of chunk IDs must be parenthesized and comma separated.

RULE options

At least one RULE option must be specified.

    Examples

    Example 1

    ALTER DISTPATH dp1 RULE FILTER CHUNKIDS (1, 2, 3)

OBJECTNAMES object-names

    Specify a rule to filter records by their object name.
    The list of object names must be parenthesized and comma separated.
    An object name must follow the following grammar: [cdb-name.]schema-
name.table-name

     Example: ALTER DISTPATH dp1 RULE FILTER OBJECTNAMES (user1.table1,
cdb1.user2.ta ble1)

    OBJECTTYPES [ DML ] [ DDL ] [ PROCEDURE ]

    Specify a rule to filter records by their object type.
    At least one object type must be specified.

    TAGS binary-tags

    Specify a rule to filter records by their tag.
    The list of tags must be parenthesized and comma separated.
    A tag must be a hexdecimal or binary value string and prefixed by
    the keywords 'HEXVALUE', 'HEXMASK', 'BINVALUE', 'BINMASK'. In case
    the tag is a BITMASK, the filter performs a bitwise AND operation
      between the mask and the tag value of an LCR record. If the result
      is equal to the MASK, then the action is applied.

    Example: ALTER DISTPATH dp1 RULE FILTER TAGS (hexvalue A4, hexvalue 18,
    hexmask F0, hexvalue F8F, binvalue 01001100, binmask 0110)

    PROCEDUREFEATURENAMES feature-names

    Specify a rule to filter records by procedure feature name.
    The list of procedure feature names must be parenthesized and comma
separated.

    Example: ALTER DISTPATH dp1 RULE FILTER PROCEDUREFEATURENAMES (RAS, AUTOCDR,
AQ)

    COLUMNVALUES column-values

    Specify a rule to filter records by their column value.
    The filtering rules must follow this grammar and be comma separated:
```

```
    [cdb-name.]schema-name.table-name.column-name ( EQ | NE | LT | GT | LE |
    GE ) column-value [ BEFORE | AFTER ]

    EQ = equal
    NE = not equal
    LT = less than
    GT = greater than
    LE = less or equal
    GE = grater or equal

    Example: ALTER DISTPATH dp1 RULE FILTER COLUMNVALUES (cdb1.user1.table1.col1
EQ
          0 BEFORE, cdb2.user2.table1.col1 GT 100)

    Example 1

    ALTER DISTPATH path1 BEGIN NOW

    Example 2

    ALTER DISTPATH path1 BEGIN SEQNO 1 RBA 10355

    Example 3

    ALTER DISTPATH path1 OPTIONS AUTORESTART RETRIES 3

    Example 4

    ALTER DISTPATH path1 RULE FILTER OBJECTNAMES (CUST1.*, CUST2.*) ACTION
EXCLUDE

    Example 5

    ALTER DISTPATH path1 RULE FILTER TAGS (AE00, MASK AB00, FF)

    Example 6

    ALTER DISTPATH path1 RULE FILTER COLUMNVALUES (user1.t1.c1 NE nope,
user1.t1.c2
    EQ 3 AFTER, user1.t2.c1 GE 5, user1.t2.c2 GT 5 BEFORE)

    Example 7-9
    ALTER DISTPATH path1 RULE FILTER OBJECTNAMES(CUST1.*,CUST2.*) RELATION AND
ACTION EXCLUDE

OPTIONS options

At least one OPTIONS option must be specified.


------------------------------------------------
ALTER EXTRACT

Use ALTER EXTRACT for the following purposes:

* To change the attributes of an Extract group created with the ADD
EXTRACT command.

* To increment a trail to the next file in the sequence.

* To upgrade to an integrated capture configuration.
```

```
* To downgrade from an integrated capture configuration.

Before using this command, stop Extract with the STOP EXTRACT group_name
command.

group-name

The name of the Extract group that is to be altered.

BEGIN {NOW | begin-datetime}
Specifies a timestamp in the data source at which to begin processing.
Timestamps
must follow the ISO8601 "2007-04-05T14:30:00Z" format.

    NOW

    NOW specifies the time at which the ADD EXTRACT command is issued.

    YYYY-MM-DDThh:mm:ssZ

    A date and time (timestamp) in the given form. For example,
2017-07-14T14:54:45Z.
    For an Oracle Extract in integrated mode, the timestamp value must be
greater
    than the timestamp at which the Extract was registered with the database.

    If using the BEGIN option, do not combine other options in the statement.

Issue separate statements, for example:

ALTER EXTRACT finance, BEGIN 2017-07-14T14:54:45Z
ALTER EXTRACT finance, ETROLLOVER
ALTER EXTRACT finance, SCN 789000

    Positioning to an LSN is precise.

EXTSEQNO sequence-number

Valid for a primary Extract in classic capture mode. Not supported for an
Oracle Extract in integrated mode. Specifies either of the following:

* sequence number of an Oracle redo log at which to begin capturing data.

* Specify the sequence number, but not any zeroes used for padding. For
example, if the trail file is ggs/var/lib/aa000026, you would specify
EXTSEQNO 26. By default, processing begins at the beginning of a trail
unless this option is used. Contact Oracle Support before using this option.

EXTRBA offset-number

Specifies the relative byte address within a transaction
log at which to begin capturing data.

SCN scn

Starts Extract at the transaction in the redo log that has the specified
Oracle system change number (SCN). This option is valid for Extract both
in classic capture and integrated modes. For Extract in integrated mode,
the SCN value must be greater than the SCN at which the Extract was
registered with the database. If using the SCN or BEGIN option for
```

```
Integrated Extract, it requires a DBLOGIN, and the SCN or timestamp
value specified cannot be below the outbound server's first SCN or timestamp.
To find the outbound server's first SCN, issue the following command:

INFO EXTRACT group-name, SHOWCH DETAIL

The first SCN value is listed as shown in the following example:

Integrated Extract outbound server first scn: 0.665884 (665884)

DESC description

A description of the distribution path.

UPGRADE INTEGRATED TRANLOG

Upgrades the Extract group from classic capture to integrated capture.
To support the upgrade, the transaction log that contains the start of
the oldest open transaction must be available on the source or downstream
mining system.

DOWNGRADE INTEGRATED TRANLOG THREADS threads-number

Downgrades the Extract group from integrated capture to classic capture.
When downgrading on a RAC system, the THREADS option must be used to specify
the number of RAC threads. On a non-RAC system, you can optionally specify
THREADS 1 to cause the downgraded classic Extract to run in threaded mode
with one thread, which is similar to doing an ADD EXTRACT with THREADS 1
on a non-RAC system.

To support the downgrade, the transaction log that contains the start
of the oldest open transaction must be available on the source or
downstream mining system.

THREADS thread-number

Valid for classic capture mode. In an Oracle RAC configuration,
alters Extract only for the specified redo thread. Only one thread
number can be specified.

ETROLLOVER

Causes Extract to increment to the next file in the trail sequence
when restarting. For example, if the current file is ET000000002 the
current file will be ET000000003 when Extract restarts. A trail can be
incremented from 000000001 through 999999999, and then the sequence numbering
starts over at 000000000.

Examples

Example 1

The following alters Extract to start processing data from July 14, 2017 at
14:54.

ALTER EXTRACT finance, BEGIN 2017-07-14T14:54:00Z

Example 2

The following alters Extract to start processing at a specific location
in the trail.
```

```
ALTER EXTRACT finance, EXTSEQNO 26, EXTRBA 338

Example 3

The following alters Extract in an Oracle RAC environment, and applies
the new begin point only for redo thread 4.

ALTER EXTRACT accounts, THREAD 4, BEGIN 2017-07-14T14:54:45Z

Example 4

The following alters Extract to increment to the next file in the
trail sequence.

ALTER EXTRACT finance, ETROLLOVER

Example 5

The following alters Extract to upgrade to integrated capture.

ALTER EXTRACT finance, UPGRADE INTEGRATED TRANLOG

Example 6

The following alters Extract to downgrade to classic capture in
a RAC environment.

ALTER EXTRACT finance, DOWNGRADE INTEGRATED TRANLOG THREADS 3

Example 7

The following alters Extract in an Oracle environment to start
processing data from source database SCN 778899.

ALTER EXTRACT finance, SCN 778899


-----------------------------------------------
ALTER EXTTRAIL

Use ALTER EXTTRAIL to change the attributes of a trail that was created
with the ADD EXTTRAIL command (a trail on the local system). The change
takes effect the next time that Extract starts.

group-name

The name of the Extract group to which the trail is bound.

MEGABYTES megabytes-number

The maximum size of a file, in megabytes. The default is 500. After using
this option, issue the SEND EXTRACT command with the ROLLOVER option to
close the current trail file and open a new one.

Example

ALTER EXTTRAIL var/lib/data/aa, EXTRACT finance, MEGABYTES 200


-----------------------------------------------
```

```
ALTER HEARTBEATTABLE

Use ALTER HEARTBEATTABLE to alter existing seed, heartbeat, and history table
options that you set with ADD HEARTBEATTABLE.

This command requires a DBLOGIN. On a CDB database, a PDB login is required.

FREQUENCY frequency-seconds-number

Alter frequency to zero (0) is equivalent to pausing the heartbeat. Heartbeat
records can be resumed by altering frequency to a value greater than 0.

RETENTION_TIME retention-days-number

Changes the heartbeat retention time specified, in days.

PURGE_FREQUENCY purge-frequency-days-number

Changes the repeat interval, in days, of the purge heartbeat table.

TARGETONLY

Modifies an existing the existing heartbeat seed and heartbeat
tables by disabling supplemental logging on both tables and it drops the
existing scheduler job for updating heartbeat table.

NOTARGETONLY

Modifies an existing the existing heartbeat seed and
heartbeat tables by enalbing supplemental logging on both tables and it
creates a  scheduler job for updating heartbeat table.

Examples

ALTER HEARTBEATTABLE FREQUENCY 60

ALTER HEARTBEATTABLE RETENTION_TIME 30

ALTER HEARTBEATTABLE PURGE_FREQUENCY 1


-----------------------------------------------
ALTER REPLICAT

Use ALTER REPLICAT to change the attributes of a Replicat group that was
created with the ADD REPLICAT command. Before using this command, stop
Replicat by issuing the STOP REPLICAT command. If this is a coordinated
Replicat group, the ALTER takes effect for all threads unless the threadID
option is used.

    Note:
    ALTER REPLICAT does not support switching from regular Replicat mode to
    coordinated mode. You must stop processes, make certain all of the enroute
    data is applied to the target, roll the trail to a new trail, drop and
    recreate the Replicat group in coordinated mode, and then start the
processes
    again.

group-name

The name of the Replicat group of a coordinated Replicat that is
```

to be altered. To specify a thread, use the full thread name, such as ALTER REPLICAT fin003, EXTSEQNO 53. If a thread ID is not specified, the ALTER takes effect for all threads of the Replicat group.

INTEGRATED

Switches Replicat from non-integrated mode to integrated mode. Transactions currently in process are applied before the switch is made.

NONINTEGRATED, CHECKPOINTTABLE table-name

Switches Replicat from integrated mode to non-integrated mode.

For CHECKPOINTTABLE, specify the owner and name of a checkpoint table. This table must be created with the ADD CHECKPOINTTABLE command before issuing ALTER EXTRACT with NONINTEGRATED.

SPECIALRUN

Creates a Replicat special run as a task. Either SPECIALRUN, EXTFILE, or EXTTRAIL is required. When Extract is in SPECIALRUN mode, do not start Replicat with the START REPLICAT command. Do not use this option with the INTEGRATED or COORDINATED option.

EXTFILE file-name

Specifies the relative or fully qualified name of an Extract file that is specified with RMTFILE in the Extract parameter file. Do not use this option with the INTEGRATED option.

EXTTRAIL trail-name

Specifies the relative or fully qualified name of a trail that was created with the ADD RMTTRAIL or ADD EXTTRAIL command.

BEGIN {NOW | begin-datetime }

Defines an initial checkpoint in the trail.

    NOW
    Begins replicating changes from the time when the group is created.

    begin-datetime
    Begins extracting changes from a specific time.

EXTSEQNO sequence-number

Specifies the sequence number of the file in a trail in which to begin processing data. Specify the sequence number, but not any zeroes used for padding. For example, if the trail file is ggs/var/lib/data/aa000026, you would specify EXTSEQNO 26.
By default, processing begins at the beginning of a trail unless this option is used. To use EXTSEQNO, you must also use EXTRBA. Contact Oracle Support before using this option.

EXTRBA offset-number

Specifies the relative byte address within the trail file that is specified by EXTSEQNO. Contact Oracle Support before using this option.

CHECKPOINTTABLE table-name | NODBCHECKPOINT

```
Not valid for Oracle GoldenGate Applications Adapter or Oracle GoldenGate for
Big
Data.

Specifies that this Replicat group will write checkpoints to the specified
table in the database. Include the owner and table name, as in hr.hr_checkpoint.
This argument overrides any default CHECKPOINTTABLE specification in the GLOBALS
file. The table must first be added with the ADD CHECKPOINTTABLE command. Do not
use this option with the INTEGRATED option.

NODBCHECKPOINT

Specifies that this Replicat group will not write checkpoints to a checkpoint
table. This argument overrides any default CHECKPOINTTABLE specification in the
GLOBALS file. This argument is required if you do not want to use a checkpoint
table with the Replicat group that is being created. Do not use this option with
the INTEGRATED option.

DESC description

Specifies a description of the group, such as 'Loads account_tab on Serv2'.
Enclose the description within quotes.

Examples

Example 1

ALTER REPLICAT sales, EXTTRAIL var/lib/data/rt

Example 2

ALTER REPLICAT sales, INTEGRATED, EXTTRAIL var/lib/data/rt

Example 3

This example alters Replicat in coordinated mode. It indicates that up
to 100 threads can be employed in parallel at any given point in processing.

ALTER REPLICAT sales, COORDINATED MAXTHREADS 100, EXTTRAIL var/lib/data/rt

------------------------------------------------
ALTER RMTTRAIL

Use ALTER RMTTRAIL to change the attributes of a trail that was created
with the ADD RMTTRAIL command (a trail on a remote system). The change takes
effect the next time that Extract starts.

trail-name

The relative or fully qualified path name of the trail. For example,
var/lib/data/aa.

group-name

The name of the Extract group to which the trail is bound.

MEGABYTES megabytes-number

The maximum size of a file, in megabytes. The default is 500. After using
this option, issue the SEND EXTRACT command with the ROLLOVER option to close
```

the current trail file and open a new one.

Example

```
ALTER RMTTRAIL var/lib/data/aa, EXTRACT finance, MEGABYTES 200
```

CD

Use CD to change the Admin Client working directory.

directory-name

The name of the directory.

CLEANUP CHECKPOINTTABLE

Not valid for Replicat for Java, Oracle GoldenGate Applications Adapter, or Oracle
GoldenGate for Big Data.

Use CLEANUP CHECKPOINTTABLE to remove checkpoint records from the checkpoint table
when there is no checkpoint file associated with it in the working Oracle GoldenGate
directory. The purpose of this command is to remove checkpoint records that are not
needed any more, either because groups were changed or files were moved.

Use the DBLOGIN command to establish a database connection before using this command.

table-name

The name of the checkpoint table to be cleaned up.

Example

```
CLEANUP CHECKPOINTTABLE ggs.fin_check
```

CLEANUP EXTRACT

Use CLEANUP EXTRACT to delete run history for the specified Extract group. The cleanup keeps the last run record intact so that Extract can resume processing from where it left off. Before using this command, stop Extract by issuing the STOP EXTRACT command.

group-name-wildcard

The name of an Extract group or a wildcard (*) to specify multiple groups. For example, T* cleans up all Extract groups whose names start with T.

SAVE save-count-number

Excludes the specified number of the most recent records from the cleanup.

Examples

```
Example 1
The following deletes all but the last record.

CLEANUP EXTRACT finance

Example 2
The following deletes all but the most recent five records.

CLEANUP EXTRACT *, SAVE 5


------------------------------------------------
CLEANUP REPLICAT

Use CLEANUP REPLICAT to delete run history for a specified Replicat group.
The cleanup keeps the last run record intact so that Replicat can resume
processing from where it left off.

Before using this command, stop Replicat by issuing the STOP REPLICAT command.

group-name-wildcard

The name of a Replicat group or a wildcard (*) to specify multiple groups. For
example, T* cleans up all Replicat groups whose names begin with T. If the
specified group (or groups) is a coordinated Replicat, the cleanup applies to
all threads.

save-count

Excludes the specified number of the most recent records from the cleanup.

Examples

Example 1

The following deletes all but the last record.

CLEANUP REPLICAT finance

Example 2

The following deletes all but the most recent five records.

CLEANUP REPLICAT *, SAVE 5

Example 3

The following deletes all but the most recent five records for thread 3
of coordinated Replicat group fin.

CLEANUP REPLICAT fin003, SAVE 5

------------------------------------------------
CONNECT

Use this to connect to an Oracle GoldenGate Service Manager to execute
other commands. You must connect to your Service Manager before you can
execute most of the Admin Client commands.

Admin client allowS connections when the server uses a self-signed certificate
```

```
though this is not the default. Admin Client does not allow connecting to a
server through HTTPS when the self-signed certificate is invalid. To override
this
behavior, use the "!" modifier with the CONNECT command.

For example, when using the Admin Client to connect to the Oracle GoldenGate
Microservices Architecture services that are secured with a self-signed
SSL certificate, you must use a command with the ! modifier:


CONNECT https://myserver.example.org as oggadmin !

server-url

The URL of the Service Manager that you want to connect to.

deployment-name

The name of the deployment that you want to connect to on the specified
Service Manager.  If only one deployment (except for Service Manager) is
defined, that deployment is the default. Otherwise, there is not a default
deployment and the DEPLOYMENT deployment-name option must be used.

proxy-uri

The URI of your proxy server in the schema://hostname[:port-number] format.

user-name

The user name for the specified Service Manager.

password

The password for the specified user name. If you do not specify the password,
you are prompted for it.

Example

connect http://prodserver.mysite.com:9700 deployment Atlanta_1 as oggadmin
password welcome1


------------------------------------------------
DBLOGIN USERIDALIAS

Use DBLOGIN to establish a database connection through Admin Client in
preparation
to issue other Oracle GoldenGate commands that affect the database. The user
who issues DBLOGIN should have the appropriate database privileges to perform
the functions that are enacted by those commands. Any other special privileges
that are required for a Admin Client command are listed in the reference
documentation
for that command.

Requirements When Configuring Extract or Replicat in Integrated Mode

If using DBLOGIN to issue ADD EXTRACT, ALTER EXTRACT, or REGISTER EXTRACT to
initiate integrated capture or ADD REPLICAT, ALTER REPLICAT, or REGISTER REPLICAT
to initiate integrated Replicat against an Oracle Database, the user who issues
DBLOGIN must:
```

```
* Have privileges granted through the Oracle
dbms_goldengate_auth.grant_admin_privilege
procedure.

* Not be changed while Extract or Replicat is in integrated mode.


Special Database Privileges to Use Log Retention in Classic Capture Mode

When in classic capture mode for an Oracle Database, Extract supports the
log-retention feature, whereby the database retains the logs that Extract
needs. To enable the log-retention feature, DBLOGIN must be issued with special
privileges before using REGISTER EXTRACT with the LOGRETENTION option. For
simplicity, you can log in as the Extract database user if the correct
privileges were granted to that user when Oracle GoldenGate was installed.
Otherwise, log in as a user with the privileges shown in the following table.

Oracle Privileges for Log Retention
Oracle EE             How to Grant Privileges version
11.1 and 11.2.0.1     1. Run package to grant Oracle GoldenGate admin privilege.
                    exec dbms_streams_auth.grant_admin_privilege('user')
                    2. Grant the 'become user' privilege.
                    grant become user to user;
11.2.0.2 and later     Run package to grant Oracle GoldenGate admin privilege.
                    exec dbms_goldengate_auth.grant_admin_privilege('user')

alias

Specifies the alias of a database user credential that is stored in the
Oracle GoldenGate credential store. To log into a pluggable database in
an Oracle multitenant container database, the user must be stored as a
connect string, such as OGGUSER@FINANCE. To log into the root container,
the user must be stored as a common user, including the C## prefix, such as
C##GGADMIN@FINANCE.

DOMAIN domain

Specifies the credential store domain for the specified alias. A valid domain
entry must exist in the credential store for the specified alias. The default
domain is OracleGoldenGate.


Examples

Example 1

DBLOGIN USERIDALIAS alias1

Example 2

DBLOGIN USERIDALIAS alias1 DOMAIN domain1


------------------------------------------------
DELETE CHECKPOINTTABLE


Not valid for Replicat for Java, Oracle GoldenGate Applications Adapter, or
Oracle
GoldenGate for Big Data.
```

```
Use DELETE CHECKPOINTTABLE to drop a checkpoint table from the database. Use the
DBLOGIN
command to establish a database connection before using this command.

To stop using a checkpoint table while the associated Replicat group remains
active,
follow these steps:

1. Run Admin Client.

2. Stop Replicat.

   STOP REPLICAT group

3. Delete the Replicat group and then add it back with the following commands.

   DELETE REPLICAT group
   ADD REPLICAT group, EXTTRAIL trail, NODBCHECKPOINT

4. Start Replicat again.

   START REPLICAT group

5. Log into the database with the DBLOGIN command, using the appropriate
authentication
options for the database.

6. Delete the checkpoint table with DELETE CHECKPOINTTABLE.

If the checkpoint table is deleted while Replicat is still running and
transactions
are occurring, Replicat will abend with an error that the checkpoint table could
not
be found. However, the checkpoints are still maintained on disk in the
checkpoint file.
To resume processing, add the checkpoint table back under the same name. Data in
the
trail resumes replicating. Then, you can delete the checkpoint table.

table-name

The name of the checkpoint table to be deleted. The schema and owner is
required.

!
Bypasses the prompt that confirms intent to delete the table.

Example

DELETE CHECKPOINTTABLE ggs.fin_check


-----------------------------------------------
DELETE CREDENTIALSTORE

Use the DELETE CREDENTIALSTORE command to remove a credential store from
the system. The credential store wallet and its contents are permanently
deleted.

The use of a credential store is not supported for the iSeries, z/OS,
and NonStop platforms. For more information about Oracle GoldenGate security
```

```
options, see Administering Oracle GoldenGate.


-----------------------------------------------
DELETE DISTPATH

Use DELETE DISTPATH to remove a distribution path.

path-name

The name of the distribution path.

Example DELETE DISTPATH path1


-----------------------------------------------

START DISTPATH ALL

Use the START DISTPATH ALL command to start all the distribution paths.

START DISTPATH ALL


-----------------------------------------------

DELETE DISTPATH ALL

Use this command to delete all the distributions paths.

DELETE DISTPATH ALL

-----------------------------------------------


DELETE EXTRACT

Use DELETE EXTRACT to delete an Extract group. This command deletes the
checkpoint file that belongs to the group, but leaves the parameter file
intact. You can then re-create the group or delete the parameter file as needed.

Before using DELETE EXTRACT, stop Extract with the STOP EXTRACT command.

To delete the trail files that are associated with the Extract group, use
the PURGE EXTTRAIL command.

group-name-wildcard

The name of an Extract group or a wildcard specification (*) to specify
multiple groups. For example, T* deletes all Extract groups whose names
start with T.

!
(Exclamation point) Deletes all Extract groups associated with a wildcard
without prompting.

-----------------------------------------------
DELETE EXTTRAIL


Use DELETE EXTTRAIL to delete the record of checkpoints associated with a
```

trail on a local system. Checkpoints are maintained in a file bearing the same name as the group in a separate sub-directory of the Oracle GoldenGate directory.

This command only deletes references to the specified trail from the checkpoint file. It does not delete the trail files themselves. To delete the trail files, use the PURGE EXTTRAIL command.

trail-name

The relative or fully qualified path name of the trail, including the two-character trail prefix.

Example

```
DELETE EXTTRAIL var/lib/data/et
```

DELETE HEARTBEATENTRY

Use DELETE HEARTBEATENTRY to delete the records in the heartbeat table with the specified process name either in the incoming or outgoing path columns. This command required a DBLOGIN. On a CDB database, a PDB login is required.

group-name

The name of the process to be cleaned.

!
(Exclamation point) Deletes all heartbeat table entries associated with a wildcard
without prompting.

DELETE HEARTBEATTABLE

Use DELETE HEARTBEATTABLE to delete tables, procedures, schedulers, and views. This command requires a DBLOGIN. On a CDB database, a PDB login is required.

!
(Exclamation point) Deletes all heartbeat tables without prompting.

DELETE MASTERKEY

Use the DELETE MASTERKEY command to mark a version of a master key for deletion. Routinely deleting older versions of a master key ensures that they cannot be used maliciously.

To view the version of a master key, use the INFO MASTERKEY command.

This command marks a version for deletion but does not physically remove it from the wallet.

> Note:
> For Oracle GoldenGate deployments using a shared wallet, the older versions of the master key should be retained after the master key

```
    is renewed until all processes are using the newest version. The
    time to wait depends on the topology, latency, and data load of
    the deployment. A minimum wait of 24 hours is a conservative estimate,
    but you may need to perform testing to determine how long it takes
    all processes to start using a new key. To determine whether all of
    the processes are using the newest version, view the report file of
    each Extract immediately after renewing the master key to confirm
    the last SCN that was mined with the old key. Then, monitor the
    Replicat report files to verify that this SCN was applied by all
    groups. At this point, you can delete the older versions of the
    master key.

See UNDELETE MASTERKEY to reverse a deletion made by DELETE MASTERKEY.

Once a version number is used, the wallet reserves it forever, and no
other key of the same version can be generated. For example, you cannot
mark version 2 of a key for deletion, then purge the wallet to remove it,
and then issue RENEW MASTERKEY to add a version 2 again. Even though only
version 1 of the key remains in the wallet after the purge, the renewal
generates version 3, not version 2.

VERSION version-number

Specifies a single version to be marked for deletion.

RANGE FROM begin-number TO end-number

Specifies a range of versions to be marked for deletion. The versions
must be contiguous. For example, specifying RANGE FROM 3 TO 6 marks
versions 3, 4, 5, and 6.

ALL

Marks all versions of the master key for deletion, including the currently
active one. When this option is used, it should always be followed by a
RENEW MASTERKEY command to create a new, current version of the master key.

Examples

Example 1

This command marks one version of the master key for deletion and returns a
message similar to the one shown.

DELETE MASTERKEY VERSION 10

Version 10 of Masterkey 'OGG_DEFAULT_MASTERKEY' deleted from wallet at
location './
var/lib'.

Example 2

This command marks versions 3, 4, 5, and 6 for deletion.

DELETE MASTERKEY RANGE FROM 3 TO 6


------------------------------------------------
DELETE PROCEDURETRANDATA

Use this to disable procedure-level supplemental logging on the database.
```

Use the DBLOGIN command to establish a database connection before using this command.

DELETE REPLICAT

Use DELETE REPLICAT to delete a Replicat group. This command deletes the checkpoint file but leaves the parameter file intact. Then you can re-create the group or delete the parameter file as needed. This command frees up trail files for purging, because the checkpoints used by the deleted group are removed (assuming no other processes are reading the file). Before using DELETE REPLICAT, stop Replicat with the STOP REPLICAT command.

If this is an integrated Replicat or a non-integrated Replicat that uses a checkpoint table, do the following after you stop Replicat:

1. Log into the database by using the DBLOGIN command. DBLOGIN enables DELETE REPLICAT to delete the checkpoints from the checkpoint table of a non-integrated Replicat or to delete the inbound server that an integrated Replicat uses. For more information, see DBLOGIN.

2. Issue DELETE REPLICAT.

group-name-wildcard

The name of a Replicat group or a wildcard (*) to specify multiple groups. For example, T* deletes all Replicat groups whose names begin with T.

!
Use this option to force the Replicat group to be deleted if the DBLOGIN command is not issued before the DELETE REPLICAT command is issued. If the group is a nonintegrated Replicat, this option deletes the group's checkpoints from the checkpoint file on disk, but not from the checkpoint table in the database. If using this option to delete an integrated Replicat group, you must use the UNREGISTER REPLICAT command to delete the inbound server from the target database. This option can also be used to ignore the prompt that occurs when a wildcard specifies multiple groups.

Note:
The basic DELETE REPLICAT command commits an existing Replicat transaction, but the ! option prevents the commit.

Example

```
DELETE REPLICAT finance
```

DELETE RMTTRAIL

Use DELETE RMTTRAIL to delete the record of checkpoints associated with a trail on a remote system. Checkpoints are maintained in a file bearing the same name as the group in a separate sub-directory of the Oracle GoldenGate directory.

This command only deletes references to the specified trail from the checkpoint file. It does not delete the trail files themselves.

trail-name

The relative or fully qualified path name of the trail, including the two-character trail prefix.

group-name

The name of the Extract group to which the trail is bound. If not specified, DELETE RMTTRAIL deletes the trail reference from all Extract groups that write to the specified trail.

Example

```
DELETE RMTTRAIL var/lib/data/et
```

DELETE SCHEMATRANDATA

Use DELETE SCHEMATRANDATA to remove the schema-level supplemental logging that was added with the ADD SCHEMATRANDATA command. Use the DBLOGIN command to establish a database connection before using this command. The user that is specified with this command must have the privilege to remove supplemental log groups.

By default, this command attempts to remove the supplemental logging of the key columns that are used by Oracle GoldenGate (can be the primary key, a unique key,
KEYCOLS columns, or all columns) and also the scheduling columns. The scheduling columns are the primary key, all of the unique keys, and all of the foreign keys.
To delete the logging of the Oracle GoldenGate key columns, but not the scheduling
columns, include the NOSCHEDULINGCOLS option with DELETE SCHEMATRANDATA. If ADD SCHEMATRANDATA was issued with the ALLCOLS option, use DELETE SCHEMATRANDATA with
the ALLCOLS option to remove the supplemental logging of all of the columns, including the Oracle GoldenGate key columns.

schema-name

The schema for which you want supplemental logging to be removed. Do not use a wildcard. If the source is an Oracle multitenant container database, make certain
to log into the pluggable database that contains the schema for which you want to remove the logging. See DBLOGIN for more information.

NOSCHEDULINGCOLS

Prevents the command from removing the supplemental logging of the scheduling columns of the tables in the specified schema. The scheduling columns are the primary key, all of the unique keys, and all of the foreign keys of a table.

ALLCOLS

Removes the supplemental logging of all of the columns of the tables in the specified schema.

Examples

Example 1

```
DELETE SCHEMATRANDATA scott
```

```
Example 2

DELETE SCHEMATRANDATA scott ALLCOLS

-----------------------------------------------
DELETE TRACETABLE

Use DELETE TRACETABLE to delete a trace table. Use the DBLOGIN command to
establish
a database connection before using this command.

table-name

The name of the trace table to be deleted.  The schema is required.

!
(Exclamation point) Deletes the trace table without prompting.

Example

DELETE TRACETABLE ora_trace


-----------------------------------------------
DELETE TRANDATA

By default, this command attempts to remove the supplemental logging of the
key columns that are used by Oracle GoldenGate (can be the primary key, a unique
key, KEYCOLS columns, or all columns) and also the scheduling columns. The
scheduling columns are the primary key, all of the unique keys, and all of the
foreign keys. To delete the logging of the Oracle GoldenGate key columns, but
not the scheduling columns, include the NOSCHEDULINGCOLS option with DELETE
TRANDATA. If ADD TRANDATA was issued with the ALLCOLS option, use DELETE TRANDATA
with the ALLCOLS option to remove the supplemental logging of all of the
columns,
including the Oracle GoldenGate key columns.
Use the DBLOGIN command to establish a database connection before using this
command. The user specified with this command must have the same privileges
that are required for ADD TRANDATA.

table-name-wildcard

The qualified name of the table. A wildcard can be used for the schema or table
name.

NOSCHEDULINGCOLS

Prevents the command from removing the supplemental logging of the scheduling
columns of the specified table. The scheduling columns are the primary key,
all of the unique keys, and all of the foreign keys of a table.

ALLCOLS

Removes the supplemental logging of all of the columns of the specified table.

Examples

Example 1

DELETE TRANDATA finance.acct
```

```
Example 2

DELETE TRANDATA finance.ac*

Example 3

DELETE TRANDATA finance.acct ALLCOLS

------------------------------------------------
DISCONNECT

Use this to disconnect from the Oracle GoldenGate Service Manager. It is not
necessary to disconnect from one Service Manager connection to connect to
another Service Manager. Use the CONNECT command to establish a connection to
a Service Manager.


------------------------------------------------
EDIT ENCKEYS

Use EDIT ENCKEYS to open the ENCKEYS file for editing in the default
text editor.


------------------------------------------------
EDIT GLOBALS

Use this to open the GLOBALS parameter file for editing in the default text
editor. The default text editor is set using the SET EDITOR command.


------------------------------------------------
EDIT PARAMS


Use EDIT PARAMS to create or change a parameter file. By default, the editor is
set with your EDITOR environment variable. You can change the default editor
with the SET EDITOR command.

file-name

Opens the specified parameter file.

Example
EDIT PARAMS finance


------------------------------------------------
ENCRYPT PASSWORD password ENCRYPTKEY encrypt-key

Use to encrypt a password that is used in an Oracle GoldenGate parameter
file or command.

password
The login password. Do not enclose the password within quotes. If the
password is case-sensitive, type it that way.

    encrypt-key

    Specifies the logical name of a user-created encryption key in a local
```

```
    ENCKEYS lookup file. The key name is used to look up the actual key in the
    ENCKEYS file. A user-created key and an associated ENCKEYS file is required
    for the AES encryption. To use encrypt-key, generate the key with KEYGEN or
    another utility, then store it in an ENCKEYS file on the source and target
systems.
    The AES ciphers have a 128-bit block size.


-----------------------------------------------
EXIT

Use to exit the Oracle GoldenGate Admin Client.


-----------------------------------------------
FLUSH SEQUENCE

Use FLUSH SEQUENCE immediately after you start Extract for the first time during
an initial synchronization or a re-synchronization. This command updates an
Oracle
sequence so that initial redo records are available at the time that Extract
starts to capture transaction data. Normally, redo is not generated until the
current cache is exhausted. The flush gives Replicat an initial start point
with which to synchronize to the correct sequence value on the target system.
From then on, Extract can use the redo that is associated with the usual cache
reservation of sequence values.

The following Oracle procedures are used by FLUSH SEQUENCE:

Database     Procedure          User and Privileges
Source         updateSequence      Grants EXECUTE to the owner of the Oracle
GoldenGate
                                DDL objects, or other selected user if not using
DDL support.
Target         replicateSequen  Grants EXECUTE to the Oracle GoldenGate Replicat
                                user.

The sequence.sqlscript installs these procedures. Normally, this script is run
as part of the Oracle GoldenGate installation process, but make certain that was
done before using FLUSH SEQUENCE. If sequence.sqlwas not run, the flush fails
and
an error message similar to the following is generated:
Cannot flush sequence {0}. Refer to the Oracle GoldenGate for Oracle
documentation
for instructions on how to set up and run the sequence.sql script. Error {1}.

2. The GLOBALS file must contain a GGSCHEMA parameter that specifies the schema
in which the procedures are installed. This user must have CONNECT, RESOURCE,
and DBA
privileges.

3. Before using FLUSH SEQUENCE, issue the DBLOGIN command as the database user
that has EXECUTE privilege on the updateSequence procedure. If logging into a
multitenant container database, log into the pluggable database that contains
the sequence that is to be flushed.

sequence-name

The name of a sequence. The schema name is required.

Example
```

```
FLUSH SEQUENCE scott.seq


------------------------------------------------
HEALTH DEPLOYMENT

Use to display the health of the specified Oracle GoldenGate deployments.

deployment-name-wildcard

The name of the deployment you are interested in. You can use an asterisk
(*) wildcard for any portion of the deployment name.

Example

HEALTH DEPLOYMENT Phoenix


------------------------------------------------
HELP

HELP

Use HELP to obtain information about an Oracle GoldenGate command. Without
additional options, HELP returns a list of commands. The command option
restricts the output to the specified command.

SHOWSYNTAX

Displays the command syntax.

command-wildcard

The command for which you want help.

Examples

Example 1

HELP add replicat

Example

To display all commands that begin with ADD.

HELP ADD


------------------------------------------------
HISTORY

Use HISTORY to view a list of the most recently issued Admin Client commands
since
the startup of the session. You can use the ! command to re-execute a command in
the list.

depth-number

Returns a specific number of recent commands, where n 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 extjd
6: info extract extjd
7: history 7

------------------------------------------------
INFO ALL

Use INFO ALL to display the status and lag (where relevant) for all Extract and
Replicat processes on a system. The basic command, without options, displays
only online (continuous) processes and Microservices Architecture services. To
display tasks, use either INFO ALL TASKS or INFO ALL ALLPROCESSES.

The Status and Lag at Chkpt (checkpoint) fields display the same process status
and lag as the INFO EXTRACT and INFO REPLICAT commands.

If Replicat is in coordinated mode, INFO ALL shows only the coordinator thread.
To view information about individual threads, use INFO REPLICAT.

TASKS

Displays information only for tasks.

ALLPROCESSES

Displays information for online processes and tasks.

Examples

Example 1

INFO ALL TASKS

Example 2

INFO ALL ALLPROCESSES

------------------------------------------------
INFO CHECKPOINTTABLE

Not valid for Replicat for Java, Oracle GoldenGate Applications Adapter, or
Oracle
GoldenGate Big Data.

Use INFO CHECKPOINTTABLE to confirm the existence of a checkpoint table and view
the
date and time that it was created. It returns a message similar to the
following:

Checkpoint table HR.CHKPT_TBLE created 2017-01-06T11:51:53.

Use the DBLOGIN command to establish a database connection before using this
command.
```

```
table-name-wildcard

The name of the checkpoint table. The schema name is required.

Example

INFO CHECKPOINTTABLE ggs.fin_check

-------------------------------------------------
INFO CREDENTIALSTORE

Use the INFO CREDENTIALSTORE command to get information about an Oracle
GoldenGate credential store. This information includes the aliases that a
credential store contains and the user IDs that correspond to them. The
encrypted passwords in the credential store are not returned.

DOMAIN domain

Returns the aliases and user IDs for a specific domain. For security purposes,
if the DOMAIN option is omitted, only the aliases and user IDs under the
default domain of OracleGoldenGateare shown. See ALTER CREDENTIALSTORE
for more information about domains.

Examples

Example 1

The following example shows the default output of INFO CREDENTIALSTORE.

INFO CREDENTIALSTORE

Example 2

The following example shows the output when DOMAIN is used.

INFO CREDENTIALSTORE DOMAIN support


-------------------------------------------------
INFO DISTPATH

Returns information about distribution paths.

ALL

Displays a list of all distribution paths with their status.

path-name

A distribution path name.

DETAIL

With DETAIL option, info distpath command displays the following additional
information for the requested distribution path.

* process and thread information.

* source database name where the data is originated
```

```
* last started timestamp and processing lag

* current and starting input and output checkpoint


-------------------------------------------------
INFO ER

Use this to get information about the specified wildcarded groups as a unit.

* The status of group (STARTING, RUNNING, STOPPED, SUSPENDED, or ABENDED).
STARTING means that the process has started but has not yet locked the
checkpoint file for processing.

* Approximate Extract lag.

* Checkpoint information.

* Process run history.

* The trail(s) to which Extract is writing.

* Status of upgrade to, or downgrade from, integrated capture. The process can
be running or stopped when INFO ER is issued. With a running process, the
status of RUNNING can mean one of the following:

* Active: Running and processing (or able to process) data. This is the
normal state of a process after it is started.

* Suspended: The process is running, but suspended due to an EVENTACTIONS
SUSPEND action. In a suspended state, the process is not active, and no data
can be processed, but the state of the current run is preserved and can be
continued by issuing the SEND command with the RESUME option.
The RBA in the INFO command reflects the last checkpointed position before
the suspend action. To determine whether the state is active or suspended,
issue the SEND command with the STATUS option.

group-name-wildcard

Name of the wildcard group.

SHOWCH checkpoints-number

The basic command shows information about the current Extract checkpoints.
Extract checkpoint positions are composed of read checkpoints in the data
source and write checkpoints in the trail. The trail type (RMTTRAIL or
EXTTRAIL) is also noted.

Optionally, specify a value for for checkpoints-number to include the
specified number of previous checkpoints as well as the current one.

Note:
You might see irregular indents and spacing in the output. This is normal
and does not affect the accuracy of the information.

DETAIL

Displays the following:

* Extract run history, including start and stop points in the data source,
expressed as a time.
```

```
* Trails to which Extract is writing.

TASKS

Displays only Extract tasks. Tasks that were specified by a wildcard
argument are not displayed by INFO EXTRACT.

ALLPROCESSES

Displays all Extract groups, including tasks.


------------------------------------------------
INFO EXTRACT

Use INFO EXTRACT to view the following information.

* The status of Extract (STARTING, RUNNING, STOPPED, or ABENDED). STARTING
means that the process has started but has not yet locked the checkpoint
file for processing.

* Approximate Extract lag.

* Checkpoint information.

* Process run history.

* The trail(s) to which Extract is writing.

* Status of upgrade to, or downgrade from, integrated capture

Extract can be running or stopped when INFO EXTRACT is issued. In the case
of a running process, the status of RUNNING can mean one of the following:

* Active: Running and processing (or able to process) data. This is the
normal state of a process after it is started.

* Suspended: The process is running, but suspended due to an EVENTACTIONS
SUSPEND action. In a suspended state, the process is not active, and no data
can be processed, but the state of the current run is preserved and can be
continued by issuing the SEND EXTRACT command with the RESUME option.
The RBA in the INFO command reflects the last checkpointed position before
the suspend action. To determine whether the state is active or suspended,
issue the SEND EXTRACT command with the STATUS option.

The basic command displays information only for online (continuous) Extract
processes. Tasks are excluded.

About Extract Lag

The Checkpoint Lagfield of the INFO EXTRACT output reflects the lag, in seconds,
at the time that the last checkpoint was written to the trail. For example,
if the following is true...

* Current time = 15:00:00

* Last checkpoint = 14:59:00

* Timestamp of the last record processed = 14:58:00
...then the lag is reported as 00:01:00 (one minute, the difference between
```

14:58
and 14:59).

A lag value of UNKNOWN indicates that the process could be running but has not yet processed records, or that the source system's clock is ahead of the target system's clock (due to clock imperfections, not time zone differences).

For more precise lag information, use LAG EXTRACT.

group-name-wildcard

The name of an Extract group or a wildcard (*) to specify multiple groups. For example, T* shows information for all Extract groups whose names start with T.

* List the PDBs registered with a specified Extract group name.

SHOWCH checkpoints-number

The basic command shows information about the current Extract checkpoints. Extract checkpoint positions are composed of read checkpoints in the data source and write checkpoints in the trail. The trail type (RMTTRAIL or EXTTRAIL) is also noted.

Optionally, specify a value for for checkpoints-number to include the specified number of previous checkpoints as well as the current one.

Note:
You might see irregular indents and spacing in the output. This is normal and does not affect the accuracy of the information.

DETAIL

Displays the following:

* Extract run history, including start and stop points in the data source, expressed as a time.

* Trails to which Extract is writing.

TASKS

Displays only Extract tasks. Tasks that were specified by a wildcard argument are not displayed by INFO EXTRACT.

ALLPROCESSES

Displays all Extract groups, including tasks.

UPGRADE | DOWNGRADE

Valid for an Oracle database only.

* UPGRADE displays whether
 the Extract can be upgraded from classic capture
mode to integrated capture mode.

* DOWNGRADE displays whether the Extract can be downgraded from integrated capture mode to classic capture mode.

If Extract cannot be upgraded or downgraded, the reason why is displayed.

A wildcarded Extract name is not allowed with this option.

Before using this command, issue the DBLOGIN command.

CONTAINERS

Lists the PDBs that are registered with the specified Extract group. However, the command errors out if it is run in non-CDB mode or the Extract group doesn't exist.

Issue the DBLOGIN command before running this command.

Examples

Example 1

INFO EXTRACT fin*, SHOWCH

Example 2

INFO EXTRACT *, TASKS

Example 3 (Oracle only)

INFO EXTRACT finance UPGRADE

INFO EXTTRAIL

Use this to get information about a local trail.

trail-name-wildcard

The name of an trail file or a wildcard (*) to specify multiple files. For example, T* shows information for all trail files whose names start with T.

INFO HEARTBEATTABLE

Use INFO HEARTBEATTABLE to display information about the heartbeat tables and options configured in the database.

This command requires a DBLOGIN. On a CDB database, a PDB login is required.

Syntax

INFO HEARTBEATTABLE

INFO MASTERKEY

Use the INFO MASTERKEY command to view the contents of the master-key wallet. The default output shows the version history of the master key, with the creation date of a version and the status of the version. The status can be one of the following:

```
* Current: Indicates this is the active version of the master key.

* Available: Indicates this version is not the current one, but can be
made active if needed.

* Deleted: Indicates that this version is marked to be deleted when the
PURGE WALLET command is issued.

version-number

Shows detailed information about a specific version of the master key.
The output includes the original creation date.


-----------------------------------------------
INFO PROCEDURETRANDATA

Use this to get information about the state of procedure-level supplemental
database logging.

Use the DBLOGIN command to establish a database connection before using this
command.


-----------------------------------------------
INFO REPLICAT

Use INFO REPLICAT to retrieve the processing history of a Replicat group.
The output of this command includes:

* The status of Replicat (STARTING, RUNNING, STOPPED or ABENDED). STARTING
means that the process has started but has not yet locked the checkpoint
file for processing.

* (Oracle database) The Replicat mode: non-integrated or integrated.

* Whether or not Replicat is in coordinated mode and, if so, how many
threads it currently uses.

* Approximate Replicat lag.

* The trail from which Replicat is reading.

* Replicat run history, including checkpoints in the trail.

* Information about the Replicat environment.


The basic command displays information only for online (continuous) Replicat
groups. Tasks are excluded.
Replicat can be stopped or running when INFO REPLICAT is issued. In the case
of a running process, the status of RUNNING can mean one of the following:

* Active: Running and processing (or able to process) data. This is the normal
state of a process after it is started.

* Suspended: The process is running, but suspended due to an EVENTACTIONS
SUSPEND action. In a suspended state, the process is not active, and no data
can be processed, but the state of the current run is preserved and can be
continued by issuing the RESUME command. The RBA in the INFO command
reflects the last checkpointed position before the suspend action. To
```

determine whether the state is active or suspended, issue the SEND REPLICAT command with the STATUS option.

About Lag

Checkpoint Lag is the lag, in seconds, at the time the last checkpoint was written to the trail. For example, consider the following example.

* Current time = 15:00:00

* Last checkpoint = 14:59:00

* Timestamp of the last record processed =14:58:00

Assuming these values, the lag is reported as 00:01:00 (one minute, the difference between 14:58 and 14:59).

A lag value of UNKNOWN indicates that Replicat could be running but has not yet processed records, or that the source system's clock is ahead of the target system's clock (due to clock imperfections, not time zone differences). For more precise lag information, use LAG REPLICAT.

group-name-wildcard

The name of:

* A Replicat group or a wildcard (*) to specify multiple groups. For example, T* shows information for all Replicat groups whose names begin with T.

SHOWCH checkpoints-number

Displays current checkpoint details, including those recorded to the checkpoint file and those recorded to the checkpoint table, if one is being used. The database checkpoint display includes the table name, the hash key (unique identifier), and the create timestamp.

Specify a value for checkpoints-number to include the specified number of previous checkpoints
as well as the current one.

DETAIL

Displays detail information. For an Oracle target, DETAIL displays the name of the inbound server when Replicat is in integrated mode.

If Replicat is in coordinated mode, DETAIL will display only the active threads. For example, if a Replicat named CR was created with a maximum of 15 threads, but only threads 7-9 are running, INFO REPLICAT group_name with DETAIL will show only the coordinator thread (CR), CR007, CR008, and CR009. Checkpoints will exist for the other threads, but they will not be shown in the command output.

TASKS

Displays only Replicat tasks. Tasks that were specified by a wildcard argument are not displayed by INFO REPLICAT.

```
ALLPROCESSES

Displays all Replicat groups, including tasks.

Examples

Example 1

INFO REPLICAT *, DETAIL, ALLPROCESSES

Example 2

INFO REPLICAT *, TASKS

Example 3

INFO REPLICAT fin003, SHOWCH


------------------------------------------------
INFO RMTTRAIL

Use INFO RMTTRAIL to retrieve configuration information for a remote trail.
It shows the name of the trail, the Extract that writes to it, the position
of the last data processed, and the assigned maximum file size.

trail-name-wildcard

A valid trail name or a wildcard (*) designating multiple trails.

Examples

Example 1

INFO RMTTRAIL *

Example 2

The following is a sample of INFO RMTTRAILoutput.
Extract Trail: /ogg/var/lib/data/aa
 Seqno Length: 9
 Flip Seqno Length: no
 Extract: OGGPMP
 Seqno: 4
 RBA: 78066
 File Size: 500M

------------------------------------------------
INFO SCHEMATRANDATA

Use INFO SCHEMATRANDATA to determine whether schema-level supplemental
logging is enabled for the specified schema or if any instantiation information
is available. Use the DBLOGIN command to establish a database connection before
using this command.

schema-name

The schema that you want supplemental logging information for. To get
information
on the appropriate schema in an Oracle multitenant container database, make
certain to log into the correct pluggable database with DBLOGIN.
```

```
Example

INFO SCHEMATRANDATA scott

-----------------------------------------------
INFO TRACETABLE

Use the INFO TRACETABLE command to verify the existence of the specified trace
table in the local instance of the database. If the table exists, Oracle
GoldenGate displays the name and the date and time that it was created;
otherwise Oracle GoldenGate displays a message stating that the table does
not exist. Use the DBLOGIN command to establish a database connection before
using this command.

table-name

The owner and name of the trace table to be verified.

Example

INFO TRACETABLE finance.ora_trace

-----------------------------------------------
INFO TRANDATA

Use INFO TRANDATA to get the following information:

Determine whether supplemental logging is enabled, and to show the
names of columns that are being logged supplementally. If all columns are being
logged, the notation ALL is displayed instead of individual column names.
Displays
any SCN instantiation information.

Use the DBLOGIN command to establish a database connection before using this
command.

table-name-wildcard

The name of the table  for which you want to view trandata information. The
schema name is required and cannot contain wildcard characters. The table name
can
contain wildcard characters.

Examples

Example 1

INFO TRANDATA finance.acct

Example 2

INFO TRANDATA finance.ac*
-----------------------------------------------
KILL ER

Use this to forcibly terminate the specified wildcarded groups as a unit.

Killing a process leaves the most recent checkpoint in place, and the current
transaction
is rolled back by the database, guaranteeing that no data is lost when the
```

```
process is
restarted. Use this command only if the process cannot be stopped gracefully
with
the STOP REPLICAT command.

group-name-wildcard

The name of the group to close. A wildcard can be used for the group name.


------------------------------------------------
KILL EXTRACT

Use KILL EXTRACT to kill an Extract process running in regular mode.
Use this command only if a process cannot be stopped gracefully with the STOP
EXTRACT command.

group-name-wildcard

The name of an Extract group or a wildcard (*) to specify multiple groups.
For example, T* kills all Extract processes whose group names start with T.

Example

KILL EXTRACT finance


------------------------------------------------
KILL REPLICAT

Use KILL REPLICAT to kill a Replicat process. Killing a process leaves the
most recent checkpoint in place, and the current transaction is rolled back
by the database, guaranteeing that no data is lost when the process is
restarted. Use this command only if Replicat cannot be stopped gracefully with
the STOP REPLICAT command.

group-name-wildcard

The name of a Replicat group or a wildcard (*) to specify multiple groups.
For example, T* kills all Replicat processes whose group names begin with T.

Example

KILL REPLICAT finance


------------------------------------------------
LAG ER

Use LAG ER to get lag information about the specified wildcarded groups.

group-name-wildcard

The name of a group or a wildcard (*) to specify multiple groups.

Example

To kill all Replicat processes whose group names begin with T.

LAG ER T*
```

```
-----------------------------------------------
LAG EXTRACT

Use LAG EXTRACT to determine a true lag time between Extract and the data
source. LAG EXTRACT calculates the lag time more precisely than INFO EXTRACT
because it communicates with Extract directly, rather than reading a
checkpoint position in the trail.

For Extract, lag is the difference, in seconds, between the time that a record
was processed by Extract (based on the system clock) and the timestamp of
that record in the data source.

If the heartbeat functionality is enable, you can view the associated lags.

group-name-wildcard

The name of an Extract group or a wildcard (*) to specify multiple groups.
For example, T* determines lag time for all Extract groups whose names
start with T.


Examples

Example 1

LAG EXTRACT *

Example 2

LAG EXTRACT *fin*


-------------------------------------------------
LAG REPLICAT

Use LAG REPLICAT to determine a true lag time between Replicat and the trail.
LAG REPLICAT estimates the lag time more precisely than INFO REPLICAT because
it communicates with Replicat directly rather than reading a checkpoint
position.

For Replicat, lag is the difference, in seconds, between the time that the last
record was processed by Replicat (based on the system clock) and the timestamp
of the record in the trail.

If the heartbeat functionality is enable, you can view the associated lags.
A DBLOGIN is required to view the heartbeat lag.

group-name-wildcard

The name of:

* A Replicat group or a wildcard (*) to specify multiple groups. For
example, T* shows lag for all Replicat groups whose names begin with T.

Examples

Example 1

LAG REPLICAT *
```

Example 2

```
LAG REPLICAT *fin*
```

LIST TABLES

Use LIST TABLES to list all tables in the database that match the specification provided with the command argument. Use the DBLOGIN command to establish a database connection before using this command. If logging into an Oracle multitenant container database, log in to the pluggable database that contains the tables that you want to list.

table-name-wildcard

The name of a table or a group of tables specified with a wildcard (*).

Example

The following shows a LIST TABLES command and sample output.

```
LIST TABLES finance.tcust*

FINANCE.TCUSTMER
FINANCE.TCUSTORD
```

MININGDBLOGIN USERIDALIAS

Supplies the alias of a database login credential. Can be used instead of the USERID option if there is a local Oracle GoldenGate credential store that contains a credential with the required privileges for this MININGDBLOGIN command. For more information about using a credential store, see Administering Oracle GoldenGate.

To log into a pluggable database in an Oracle multitenant container database, the user must be stored as a connect string, such as OGGUSER@FINANCE. To log into the root container, the user must be stored as a common user, including the C## prefix, such as C##GGADMIN@FINANCE. For more information about configuring Oracle GoldenGate for a CDB, see Using Oracle GoldenGate for Oracle Database.

alias

Specifies the alias of a database user credential that is stored in the Oracle GoldenGate credential store. The user that is specified with USERIDALIAS must be the common database user.

DOMAIN domain

Specifies the credential store domain for the specified alias. A valid domain entry must exist in the credential store for the specified alias.

NOALLOWNESTED

Use the NOALLOWNESTED command to disable the use of nested OBEY files. A nested OBEY file is one that references another OBEY file.

When you exit your Admin Client session, the next Admin Client session will revert to NOALLOWNESTED.

This is the default. An attempt to run a nested OBEY file in the default mode of NOALLOWNESTED will cause an error that is similar to the following:

ERROR: Nested OBEY scripts not allowed. Use ALLOWNESTED to allow nested scripts.

OBEY

Use OBEY to process a file that contains a list of Oracle GoldenGate commands. OBEY is useful for executing commands that are frequently used in sequence.

You can call one OBEY file from another one. This is called a nested OBEY file. To use nested OBEY files, you must enable the functionality by first issuing the ALLOWNESTED command. There is no limit to nesting levels. See ALLOWNESTED | NOALLOWNESTED.

file-name

The relative or fully qualified path name of the file that contains the list of commands.

Examples

Example 1

```
OBEY ./mycommands.txt
```

The preceding command executes a file that looks similar to the following example:

```
sbalouse: This text is not correctly formatted and should read:

add extract fin, tranlog, begin now
add exttrail ggs/var/lib/data/aa, extract fin
add extract hr, tranlog, begin now
add exttrail ggs/var/lib/data/bb, extract hr
add replicat fin2, exttrail ggs/var/lib/data/aa, begin now
add replicat hr2, exttrail ggs/var/lib/data/bb, begin now
obey ./startcmds.txt
```

Example 2

The following example illustrates a nested OBEY file. Assume an OBEY file named addcmds.txt. Inside this file, there is another OBEY command that calls the OBEY file named startcmds.txt, which executes another set of commands.

```
OBEY ./addcmds.txt

(This OBEY statement executes the following:)

add extract fin, tranlog, begin now add exttrail ggs/var/lib/data/aa, extract
fin add
extract hr, tranlog, begin now add exttrail ggs/var/lib/data/bb, extract hr add
replicat
fin2, exttrail ggs/var/lib/data/aa, begin now add replicat hr2, exttrail
ggs/var/lib/data/bb,
begin now obey ./startcmds.txt
```

```
(The nested startcmds.txt file executes the following:)
start extract *
info extract *, detail
start replicat *
info replicat *, detail


------------------------------------------------
PURGE EXTTRAIL

Use PURGE EXTTRAIL to remove files related to a local trail from the file system.

trail-name

The relative or fully qualified path name of the trail.

!

Bypasses the prompt that confirms intent to delete the trail file.


------------------------------------------------
PURGE WALLET

Use the PURGE WALLET command to permanently remove master key versions
from the master-key wallet. Only the versions that are marked for deletion
by the DELETE MASTERKEY command are removed. The purge is not reversible.

Note:
For Oracle GoldenGate deployments using a shared wallet, the older versions
of the master key should be retained after the master key is renewed until
all processes are using the newest version. The time to wait depends on the
topology, latency, and data load of the deployment. A minimum wait of 24
hours is a conservative estimate, but you may need to perform testing to
determine how long it takes for all processes to start using a new key.
determine whether all of the processes are using the newest version,
view the report file of each Extract immediately after renewing the master
to confirm the last SCN that was mined with the old key. Then, monitor
the Replicat report files to verify that this SCN was applied by all
Replicat groups. At this point, you can delete the older versions of
the master key.

After purging a wallet that is not maintained centrally on shared storage,
the updated wallet can be copied to all of the other systems in the Oracle
GoldenGate configuration that use this wallet, so that no purged keys
remain in the configuration. Before doing so, Extract must be stopped and
then all of the downstream Oracle GoldenGate processes must be allowed to
finish processing their trails and then be stopped. After the wallet is
copied into place, the processes can be started again. For detailed
instructions, see Administering Oracle GoldenGate.


------------------------------------------------
REGISTER EXTRACT

Use REGISTER EXTRACT to register a primary Extract group with an Oracle
Database to:

* Enable integrated capture mode

* Specify options for integrated capturing from a multitenant container database
```

```
* Enable Extract in classic capture mode to work with Oracle Recovery Manager
to retain the archive logs needed for recovery

To unregister an Extract group from the database, use the UNREGISTER EXTRACT
command.

group-name

The name of the Extract group that is to be registered. Do not use a wildcard.

LOGRETENTION

Enables an Extract group in classic capture mode to work with Oracle Recovery
Manager (RMAN) to retain the logs that Extract needs for recovery. LOGRETENTION
is ignored if the Extract group is configured for integrated capture.

DATABASE [
CONTAINER (container-list |
ADD CONTAINER container-list |
DROP CONTAINER container-list
]

Without options, DATABASE enables integrated capture from a non-CDB database
for the Extract group. In this mode, Extract integrates with the database
logmining server to receive change data in the form of logical change records
(LCR). Extract does not read the redo logs. Extract performs capture processing,
transformation, and other requirements. The DML filtering is performed by the
Logmining server. For support information and configuration steps, see
Using Oracle GoldenGate for Oracle Database.

Before using REGISTER EXTRACT with DATABASE, use the DBLOGIN command for all
extracts with the privileges granted using the
dbms_goldengate_auth.grant_admin_privilege
procedure. If you have a downstream configuration, then you must also issue the
MININGDBLOGIN command. If the source database you are registering is a CDB
database
and Extract will fetch data, then grant_admin_privilegemust be called with the
CONTAINER='ALL' parameter.

Before using REGISTER EXTRACT, use ADD EXTRACT with the INTEGRATED TRANLOG
option
to create an Extract group of the same name. You must add an Extract group
before registering it.

    CONTAINER container-list

    Applies the registration to a list of one or more pluggable databases
(containers)
    of a multitenant container database (CDB). Specify one or more pluggable
databases
    as a comma-delimited list within parentheses, for example: CONTAINER (pdb1,
pdb2,
    pdb3). All of the pluggable databases must exist in the database and all
names
    can be explicit or wildcarded.

    ADD CONTAINER container-list

    Adds the specified pluggable database to an existing Extract capture
configuration.
```

```
    Specify one or more pluggable databases as a comma-delimited list within
    parentheses, for example: ADD CONTAINER (pdb1, pdb2, pdb3). Before issuing
    REGISTER EXTRACT with this option, stop the Extract group.

    Adding containers at particular SCN on an existing Extract is not supported.

    DROP CONTAINER container-list

    Drops the specified pluggable database from an existing Extract capture
    configuration. Specify one or more pluggable databases as a comma-delimited
    list within parentheses, for example: DROP CONTAINER (pdb1, pdb2, pdb3).
    Before issuing REGISTER EXTRACT with this option, stop the Extract group.

LOGRETENTION

LOGRETENTION creates an underlying Oracle Streams capture process that is
dedicated
to the Extract group and has a similar name. This capture is used only for the
purpose of log retention.

The logs are retained from the time that REGISTER EXTRACT is issued, based on
the
current database SCN. The log-retention feature
 is controlled with the LOGRETENTION
option of the TRANLOGOPTIONS parameter.

Before using REGISTER EXTRACT with LOGRETENTION, issue the DBLOGIN command with
the privileges shown in DBLOGIN.

SCN scn

Registers Extract to begin capture at a specific system change number (SCN)
in the past. Without this option, capture begins from the time that REGISTER
EXTRACT
is issued. The specified SCN must correspond to the begin SCN of a dictionary
build operation in a log file. You can issue the following query to find all
valid SCN values:

SELECT first_change#
  FROM v$archived_log
WHERE dictionary_begin = 'YES' AND
  standby_dest = 'NO' AND
  name IS NOT NULL AND
  status = 'A';

When used alone, the SCN value is the beginning SCN of the dictionary build
operation in a log file.

When used in conjunction with SHARE AUTOMATIC or SHARE extract_name, then the
specified SCN is the start_scn for the capture session and has the following
restrictions:

* Should be lesser than or equal to the current SCN.

* Should be greater than the minimum (first SCN) of the existing captures.


SHARE [
AUTOMATIC |

group-name|
```

```
NONE]}

Registers the extract to return to an existing LogMiner data dictionary build
with a specified SCN creating a clone. This allows for faster creation captures
by leveraging existing dictionary builds.

SHARE cannot be used on a CDB.

The following commands are supported:

REGISTER EXTRACT extract database SCN #### SHARE AUTOMATIC
REGISTER EXTRACT extract database SHARE NONE
REGISTER EXTRACT extract database SCN #### SHARE NONE

Or

REGISTER EXTRACT extract DATABASE SHARE NONE
REGISTER EXTRACT extract DATABASE SCN #### SHARE NONE

In contrast, the following commands are not supported in a downstream
configuration:

REGISTER EXTRACT extract DATABASE SHARE AUTOMATIC

    AUTOMATIC
    Clone from the existing closest capture. If no suitable clone candidate
    is found, then a new build is created.

    group-name
    The name of the group.

    NONE
    Does not clone or create a new build; this is the default.

In a downstream configuration, the SHARE clause must be used in conjunction
with the SCN clause when registering for Extract.

Examples

Example 1

REGISTER EXTRACT sales LOGRETENTION

Example 2

REGISTER EXTRACT sales DATABASE

Example 3

REGISTER EXTRACT sales DATABASE CONTAINER (sales, finance, hr)

Example 4

REGISTER EXTRACT sales DATABASE ADD CONTAINER (customers)

Example 5

REGISTER EXTRACT sales DATABASE DROP CONTAINER (finance)

Example 6
```

```
REGISTER EXTRACT sales DATABASE SCN 136589

The beginning SCN of the dictionary build is 136589.

Example 7

REGISTER EXTRACT sales DATABASE SCN 67000 SHARE ext2

The valid start SCN, 67000 in this case; it is not necessarily the
current SCN.

Example 8

REGISTER EXTRACT sales DATABASE CONTAINER (sales, finance, hr)
SCN 136589

-----------------------------------------------
REGISTER REPLICAT

Use the REGISTER REPLICAT command to register a Replicat group with a
target Oracle Database to support integrated Replicat mode. This command
should not be necessary under normal Replicat conditions. The startup
registers Replicat with the target database automatically. Use this
command only if Oracle GoldenGate returns a message that an integrated
Replicat is not registered with the database.

Before issuing this command, issue the DBLOGIN command as the Replicat
database user with privileges granted through
dbms_goldengate_auth.grant_admin_privilege.

group-name-wildcard

A Replicat group or a wildcard (*) to specify multiple groups. For
example, T* registers all Replicat groups whose names begin with T.

DATABASE

Required keyword to register with the target database. Creates a
database inbound server and associates it with the specified Replicat group.

Example

REGISTER REPLICAT sales DATABASE


-----------------------------------------------
RENEW MASTERKEY

Use the RENEW MASTERKEY command to create a new version of the master
encryption key in the master-key wallet. All versions of a master key remain
in the wallet until they are marked for deletion with the DELETE MASTERKEY
command and then the wallet is purged with the PURGE WALLET command.

After renewing a master key in a wallet that is not maintained centrally on
shared storage, the updated wallet must be copied to all of the other systems
in the Oracle GoldenGate configuration that use this wallet. Before doing so,
Extract must be stopped and then all of the downstream Oracle GoldenGate
processes must be allowed to finish processing their trails and then be
stopped. After the wallet is copied into place, the processes can be started
again. For detailed instructions, see Administering Oracle GoldenGate.
```

```
-----------------------------------------------
RESTART DEPLOYMENT


Use RESTART DEPLOYMENT to restart the specified deployment.

deployment-name-wildcard

The name of the deployment or a wildcard (*) to specify multiple deployments.
For example, P*restarts all deployments whose names start with P.

Example

RESTART DEPLOYMENT Phoenix


-----------------------------------------------
RESTART ER

Use RESTART ER to stop then start the specified wildcarded groups. ER processes
that are already stopped are started.

group-name-wildcard

The name of the group or a wildcard (*) to specify multiple groups.
For example, T* restarts all groups whose names start with T.


-----------------------------------------------
RESTART EXTRACT

Use RESTART EXTRACT to stop then start an Extract group.

group-name-wildcard

The name of an Extract group or a wildcard (*) to specify multiple groups. For
example, T* restarts all Extract groups whose names begin with T.

ATCSN csn | AFTERCSN csn

Specifies an alternate start point.

    ATCSN
    Directs Extract to position its restart point at the first transaction that
    has the specified CSN. Any transactions in the data source that have CSN
    values less than the specified one are skipped.

    AFTERCSN
    Directs Extract to position its restart point at the beginning of the first
    transaction after the one that has the specified CSN. Any transactions in
    the data source that have CSN values that are less than, or equal to, the
    specified one are skipped.

    csn
    Specifies a CSN value. Enter the CSN value in the format that is valid for
    the database. Extract abends if the format is invalid and writes a message
    to the report file. To determine the CSN to supply after an initial load is
    complete, use the serial identifier at which the load utility completed.
    Otherwise, follow the instructions in the initial load procedure for
    determining when to start Extract.
```

The following are additional guidelines to observe when using ATCSN and AFTERCSN:

* The CSN is stored in the file header so that it is available to downstream processes.

* When a record that is specified with a CSN is found, Extract issues a checkpoint.
The checkpoint ensures that subsequent Extract startups begin from the requested location, and not from a point prior to the requested CSN.

* You must establish a physical start point in the transaction log or trail for Extract with ADD EXTRACT or ALTER EXTRACT before using ATCSN or AFTERCSN. These options are intended to be an additional filter after Extract is positioned to a physical location in the data source.

Examples

Example 1

```
RESTART EXTRACT finance
```

Example 2

```
RESTART EXTRACT finance ATCSN 684993
```

Example 3

```
RESTART EXTRACT finance AFTERCSN 684993
```

RESTART REPLICAT

Use RESTART REPLICAT to stop then start a Replicat group. To confirm that Replicat has started, use the INFO REPLICAT or STATUS REPLICAT command.

Normal Starting Point

Replicat can be restarted at its normal start point (from initial or current checkpoints) or from an alternate, user-specified position in the trail.

RESTART REPLICAT, without any options, causes Replicat to start processing at one of the following points to maintain data integrity:

* After graceful or abnormal termination: At the first unprocessed transaction in the trail from the previous run, as represented by the current read checkpoint.

* First-time startup after the group was created: From the beginning of the active trail file (seqno 0, rba 0).

Alternate Starting Point

The SKIPTRANSACTION, ATCSN, and AFTERCSN options of RESTART REPLICAT cause Replicat as a whole, or specific threads of a coordinated Replicat, to begin processing at a transaction in the trail other than the normal start point. Use these options to:

* Specify a logical recovery position when an error prevents Replicat from

```
moving forward in the trail. Replicat can be positioned to skip the
offending transaction or transactions, with the understanding that the
data will not be applied to the target.

* Skip replicated transactions that will cause duplicate-record and missing-
record
errors after a backup is applied to the target during an initial load. These
options cause Replicat to discard transactions that occurred earlier than the
most recent set of changes that were captured in the backup. You can map the
value of the serial identifier that corresponds to the completion of the backup
to a CSN value, and then start Replicat to begin applying transactions from
the specified CSN onward.

group-name-wildcard

The name of a Replicat group or a wildcard (*) to specify multiple groups. For
example, T* restarts all Replicat groups whose names begin with T.

SKIPTRANSACTION

Causes Replicat to skip the first transaction after its expected startup
position in the trail. All operations from that first transaction are excluded.

If the MAXTRANSOPS parameter is also being used for this Replicat, it is
possible
that the process will start to read the trail file from somewhere in the middle
of a transaction. In that case, the remainder of the partial transaction is
skipped
and Replicat resumes normal processing from the next begin-transaction record
in the file. The skipped records are written to the discard file if the
DISCARDFILE
parameter is being used; otherwise, a message is written to the report file that
is similar to:

User requested RESTART SKIPTRANSACTION. The current transaction will be skipped.
Transaction ID txid, position Seqno seqno, RBA rba

SKIPTRANSACTION is valid only when the trail that Replicat is reading is part of
an online change synchronization configuration (with checkpoints). Not valid
for task-type initial loads (where SPECIALRUN is used with ADD REPLICAT).

    ATCSN csn| AFTERCSN csn
    Sets a user-defined start point at a specific CSN. When ATCSN or AFTERCSN is
    used, a message similar to one of the following is written to the report
file:

    User requested start at commit sequence number (CSN) csn-string

    User requested start after commit sequence number (CSN) csn-string

    General information about these options:

    * Valid only when the trail that Replicat is reading is part of an online
change
    synchronization configuration (with checkpoints). Not valid for task-type
initial
    loads (where SPECIALRUN is used with ADD REPLICAT).

    * To support starting at, or after, a CSN, the trail must be of Oracle
GoldenGate
    version 10.0.0 or later, because the CSN is stored in the first trail record
```

```
of
    each transaction. If Replicat is started with AFTERCSN against an earlier
trail
    version, Replicat will abend and write an error to the report stating that
the
    trail format is not supported.

    ATCSN

    Causes Replicat to start processing at the transaction that has the
specified
    CSN. Any transactions in the trail that have CSN values that are less than
    the specified one are skipped.

    AFTERCSN

    Causes Replicat to start processing at the transaction that occurred after
    the one with the specified CSN. Any transactions in the trail that have CSN
    values that are less than, or equal to, the specified one are skipped.

    csn

    Specifies a CSN value. Enter the CSN value in the format that is valid for
    the database. Replicat abends if the format is invalid and writes a message
    to the report file. To determine the CSN to supply after an initial load is
    complete, use the commit identifier at which the load utility completed the
    load. Otherwise, follow the instructions in the initial load procedure for
    determining when to start Replicat.

FILTERDUPTRANSACTIONS | NOFILTERDUPTRANSACTIONS

Causes Replicat to ignore transactions that it has already processed. Use when
Extract was repositioned to a new start point (see the ATCSN or AFTERCSN option
of
START EXTRACT) and you are confident that there are duplicate transactions
in the trail that could cause Replicat to abend. This option requires the use
of a checkpoint table. If the database is Oracle, this option is valid only
for Replicat in nonintegrated mode. In case of Integrated mode and automatic
target trail file regeneration, the Integrated mode handles the duplicate
transactions transparently. The default is FILTERDUPTRANSACTIONS.

THREADS thread-list

Valid for SKIPTRANSACTION, ATCSN, and AFTERCSN when Replicat is in coordinated
mode. Not valid for RESTART REPLICAT without those options. Starts the specified
Replicat thread or threads at the specified location.

    thread-list

    A comma-delimted list of ranges in the format of threadIDlow-threadIDhigh,
    threadIDlow-threadIDhigh.

!

(Exclamation point) Restarts Replicat immediately. The transaction is stopped.

Examples

Example 1

RESTART REPLICAT finance
```

```
Example 2

The following restarts Replicat at a CSN.

RESTART REPLICAT finance, ATCSN 6488359

Example 3

The following causes threads 4 and 5 of a coordinated Replicat to skip
the first transaction after their last checkpoint when Replicat is started.
If this were a 10-thread coordinated Replicat, threads 0-3 and 6-10 would
all restart at the normal start point, that of their last checkpoint.

RESTART REPLICAT fin SKIPTRANSACTION THREADS(4-5)

Example 4

The following example causes threads 1-3 of a coordinated Replicat to
restart at CSN 6488359, threads 9-10 to start after CSN 6488360, and threads
7 and 8 to skip the first transaction after its last checkpoint.

RESTART REPLICAT fin ATCSN 6488359 THREADS(1-3), AFTERCSN 6488360
THREADS(9-10), SKIPTRANSACTION THREADS(7,8)

-----------------------------------------------
RESTART SERVICE

Use RESTART SERVICE to restart the specified Oracle GoldenGate services.

service-name-wildcard

The name of an service or a wildcard (*) to specify multiple services. Valid
services are ADMINSRVR, DISTSRVR, RECVSRVR, and PMSRVR.

Example

RESTART SERVICE ADMIN*


-----------------------------------------------

SEND ER

Use SEND ER to send instructions to, or returns information about, the specified
wildcarded groups.

group-name-wildcard

Name of the group or a wildcard (*) to specify multiple groups. For
example, T* sends commands to all groups whose names begin with T.


-----------------------------------------------
SEND EXTRACT


Use SEND EXTRACT to communicate with a running Extract process. The
request is processed as soon as Extract is ready to accept commands from users.

group-name-wildcard
```

```
The name of the Extract group or a wildcard (*) to specify multiple
groups. For example, T* sends the command to all Extract processes whose
group names start with T. If an Extract is not running, an error is returned.

command

The command for the Extract process.


-----------------------------------------------
SEND REPLICAT

Use SEND REPLICAT to communicate with a starting or running Replicat process.
The request is processed as soon as Replicat is ready to accept commands
from users.

group-name-wildcard

The name of the Replicat group or a wildcard (*) to specify multiple
groups. For example, T* sends the command to all Replicat processes whose
group names start with T. If an Replicat is not running, an error is returned.

command

The command for the Replicat process.

DEPENDENCYINFO|DEPINFO [TXNCOUNT num]

Prints out information from the PR transaction dependency graph.
First, it shows the transaction groups currently being executed and then
the transactions waiting on some other due to a dependency.

The following example prints the dependency information for the Rep3
Replicat:

OGG (... demo) 6> send replicat rep3 depinfo

Sending depinfo request to REPLICAT REP3 ...

Scheduler 0:
Transaction groups currently being executed:
Group 0:0.3.32.1595, 0.2.12.1720, 0.8.23.1690, 0.6.18.1871
Group 1:0.8.15.1692, 0.1.13.1285, 0.10.17.1319, 0.3.18.1595
Group 2:0.4.26.1158, 0.8.11.1690, 0.5.8.1580, 0.9.17.1660
Group 3:0.5.4.1587, 0.2.10.1693, 0.9.9.1670
Waiting transactions:
Transaction with XID 0.6.29.1872 is waiting on transaction with XID
0.5.23.1583
Transaction with XID 0.8.2.1693 is waiting on transaction with XID
0.9.30.1668
Transaction with XID 0.10.16.1312 is waiting on transaction with XID
0.3.11.1592
Transaction with XID 0.2.15.1695 is waiting on transaction with XID
0.9.18.1664
Transaction with XID 0.9.8.1631 is waiting on transaction with XID
0.7.20.1187
Transaction with XID 0.1.28.1290 is waiting on transaction with XID
0.4.30.1156
Transaction with XID 0.5.13.1582 is waiting on transaction with XID
0.8.14.1689
```

```
Transaction with XID 0.3.12.1597 is waiting on transaction with XID
0.7.31.1184
Transaction with XID 0.10.4.1319 is waiting on transaction with XID
0.6.13.1873
Transaction with XID 0.4.10.1152 is waiting on transaction with XID
0.7.19.1187


------------------------------------------------
SET DEBUG

Use SET DEBUG  to enable or disable debugging mode for the Admin Client. By
default, this is set by the value of the environment variable, ADMINCLIENT_DEBUG.

Use the SHOW command to see the value of the SET DEBUG variable.

ON

Debugging mode is enabled.

OFF

Debugging mode is disabled. This is the default.


------------------------------------------------

SET EDITOR

Use SET EDITOR to change the default text editor for the current session of
Admin Client. The default editors are Notepad for Windows and vi for UNIX and
Linux. By default, this is set by the value of the environment variable, EDITOR.

Use the SHOW command to see the value of the SET EDITOR variable.

command

Any text editor.

Example

The following example changes the default editor to Wordpad.

SET EDITOR wordpad


------------------------------------------------
SET PAGER

Use SET PAGER to set the default text viewer program for viewing parameter and
report files. By default, this is set by the value of the environment variable,
PAGER;
on UNIX and Linux is defaults to less and on more on Windows.

Use the SHOW command to see the value of the SET PAGER variable.

command

Any text viewer.
```

```
-----------------------------------------------
SHELL

Use SHELL to execute shell commands from within the interface. This
command is run on the local system and not on the system where the
Administration Server or Service Manager is running.

command

The system command to execute.

Example

SHELL ls -l *.obey


-----------------------------------------------
SHOW

Use SHOW to display the Oracle GoldenGate environment variables.

Example

Following is sample output for this command:

Current directory: /scratch/ogg/sa/bin
DEBUG        : OFF
EDITOR        : vi
PAGER        : more


-----------------------------------------------
START DEPLOYMENT

Use START DEPLOYMENT to start the specified Oracle GoldenGate deployments.

deployment-name-wildcard

The name of the deployment or a wildcard (*) to specify multiple
deployments. For example, T* sends the command to all deployments whose
group names start with T.


-----------------------------------------------
START DISTPATH

Use START DISTPATH  to start a distribution path. To confirm that the
distribution path has started, use the INFO DISTPATH command. To change the
distribution path start point, use the ALTER DISTPATH command.

path-name

The distribution path name.


-----------------------------------------------

START ER

Use START ER  to start the specified wildcarded groups.
```

```
group-name-wildcard

The name of the Extract or Replicat group or a wildcard (*) to specify multiple
groups. For example, T* starts all processes whose group names start with T.


-----------------------------------------------
START EXTRACT

Use START EXTRACT to start one or more Extract process. To confirm that Extract
has started, use the INFO EXTRACT or STATUS EXTRACT command. Extract can be
started at its normal start point (from initial or current checkpoints) or
from an alternate, user-specified position in the data source.

Normal Starting Point

Without options, START EXTRACT directs a primary Extract to start processing at
one of the following locations in the data source to maintain data integrity:

* After graceful or abnormal termination: At the first unprocessed transaction
in the data source from the previous run, as represented by the current read
checkpoint.

* First-time startup after the group was created: At the start point specified
with the ADD EXTRACT command.

Alternate Starting Point

Before starting Extract with ATCSN or AFTERCSN, you must establish a physical
starting location with one of the following commands:

* ADD EXTRACT with the BEGIN option set to a timestamp that is earlier than the
CSN value specified with ATCSN or AFTERCSN. The transaction log that contains
the timestamp and every log thereafter must be available on the system before
Extract is started.

* ALTER EXTRACT to the sequence number of the log that contains the CSN
specified with ATCSNor AFTERCSN.

group-name-wildcard

The name of an Extract group or a wildcard (*) to specify multiple groups. For
example, T* starts all Extract groups whose names begin with T.

ATCSN csn | AFTERCSN csn

Specifies an alternate start point.

    ATCSN
    Directs Extract to position its start point at the first transaction that
    has the specified CSN. Any transactions in the data source that have CSN
    values less than the specified one are skipped.

    AFTERCSN
    Directs Extract to position its start point at the beginning of the first
    transaction after the one that has the specified CSN. Any transactions in
    the data source that have CSN values that are less than, or equal to, the
    specified one are skipped.

    csn
    Specifies a CSN value. Enter the CSN value in the format that is valid for
```

the database. Extract abends if the format is invalid and writes a message to the report file. To determine the CSN to supply after an initial load is complete, use the serial identifier at which the load utility completed. Otherwise, follow the instructions in the initial load procedure for determining when to start Extract.

The following are additional guidelines to observe when using ATCSN and AFTERCSN:

* The CSN is stored in the file header so that it is available to downstream processes.

* When a record that is specified with a CSN is found, Extract issues a checkpoint.
The checkpoint ensures that subsequent Extract startups begin from the requested location, and not from a point prior to the requested CSN.

* You must establish a physical start point in the transaction log or trail for Extract with ADD EXTRACT or ALTER EXTRACT before using ATCSN or AFTERCSN. These options are intended to be an additional filter after Extract is positioned to a physical location in the data source.

Examples

Example 1

START EXTRACT finance

Example 2

START EXTRACT finance ATCSN 684993

Example 3

START EXTRACT finance AFTERCSN 684993

START REPLICAT

Use START REPLICAT to start one or more Replicat processes. To confirm that Replicat
has started, use the INFO REPLICAT or STATUS REPLICAT command.

When starting an integrated Replicat group for an Oracle target database, START REPLICAT automatically registers Replicat with the target database.

A coordinated Replicat can only be started as a whole. There is no option to start individual threads. If the prior shutdown of a coordinated Replicat was not clean, the threads may have stopped at different positions in the trail file. If this happens, START REPLICAT writes a warning if the parameter file was changed since the prior run and raises an error if the number of threads was changed. To resolve these problems and start Replicat again, see Administering Oracle GoldenGate.

Normal Starting Point

Replicat can be started at its normal start point (from initial or current checkpoints) or from an alternate, user-specified position in the trail.

START REPLICAT, without any options, causes Replicat to start processing

at one of the following points to maintain data integrity:

* After graceful or abnormal termination: At the first unprocessed transaction in the trail from the previous run, as represented by the current read checkpoint.

* First-time startup after the group was created: From the beginning of the active trail file (seqno 0, rba 0).

Alternate Starting Point

The SKIPTRANSACTION, ATCSN, and AFTERCSN options of START REPLICAT cause Replicat as a whole, or specific threads of a coordinated Replicat, to begin processing at a transaction in the trail other than the normal start point. Use these options to:

* Specify a logical recovery position when an error prevents Replicat from moving forward in the trail. Replicat can be positioned to skip the offending transaction or transactions, with the understanding that the data will not be applied to the target.

* Skip replicated transactions that will cause duplicate-record and missing-record
errors after a backup is applied to the target during an initial load. These options cause Replicat to discard transactions that occurred earlier than the most recent set of changes that were captured in the backup. You can map the value of the serial identifier that corresponds to the completion of the backup to a CSN value, and then start Replicat to begin applying transactions from the specified CSN onward.

Note:
Skipping a transaction, or starting at or after a CSN, might cause Replicat to start more slowly than normal, depending on how much data in the trail must be read before arriving at the appropriate transaction record. To view the startup progress, use the SEND REPLICAT command with the STATUS option. To omit the need for Replicat to read through transactions that ultimately will be skipped, you can use the ATCSN or AFTERCSN option when starting Extract, so that those transactions are omitted from the trail. See START EXTRACT.

group-name-wildcard

The name of a Replicat group or a wildcard (*) to specify multiple groups. For example, T* starts all Replicat groups whose names begin with T.

SKIPTRANSACTION

Causes Replicat to skip the first transaction after its expected startup position in the trail. All operations from that first transaction are excluded.

If the MAXTRANSOPS parameter is also being used for this Replicat, it is possible
that the process will start to read the trail file from somewhere in the middle of a transaction. In that case, the remainder of the partial transaction is skipped
and Replicat resumes normal processing from the next begin-transaction record in the file. The skipped records are written to the discard file if the DISCARDFILE
parameter is being used; otherwise, a message is written to the report file that is similar to:

```
User requested START SKIPTRANSACTION. The current transaction will be skipped.
Transaction ID txid, position Seqno seqno, RBA rba

SKIPTRANSACTION is valid only when the trail that Replicat is reading is part of
an online change synchronization configuration (with checkpoints). Not valid
for task-type initial loads (where SPECIALRUN is used with ADD REPLICAT).

ATCSN csn| AFTERCSN csn
Sets a user-defined start point at a specific CSN. When ATCSN or AFTERCSN is
used, a message similar to one of the following is written to the report file:

User requested start at commit sequence number (CSN) csn-string

User requested start after commit sequence number (CSN) csn-string

General information about these options:

* Valid only when the trail that Replicat is reading is part of an online change
synchronization configuration (with checkpoints). Not valid for task-type
initial
loads (where SPECIALRUN is used with ADD REPLICAT).

* To support starting at, or after, a CSN, the trail must be of Oracle
GoldenGate
version 10.0.0 or later, because the CSN is stored in the first trail record of
each transaction. If Replicat is started with AFTERCSN against an earlier trail
version, Replicat will abend and write an error to the report stating that the
trail format is not supported.

    ATCSN

    Causes Replicat to start processing at the transaction that has the
specified
    CSN. Any transactions in the trail that have CSN values that are less than
    the specified one are skipped.

    AFTERCSN

    Causes Replicat to start processing at the transaction that occurred after
    the one with the specified CSN. Any transactions in the trail that have CSN
    values that are less than, or equal to, the specified one are skipped.

    csn

    Specifies a CSN value. Enter the CSN value in the format that is valid for
    the database. Replicat abends if the format is invalid and writes a message
    to the report file. To determine the CSN to supply after an initial load
    is complete, use the commit identifier at which the load utility completed
    the load. Otherwise, follow the instructions in the initial load procedure
    for determining when to start Replicat.

FILTERDUPTRANSACTIONS | NOFILTERDUPTRANSACTIONS

Causes Replicat to ignore transactions that it has already processed. Use when
Extract was repositioned to a new start point (see the ATCSN or AFTERCSN option
of
START EXTRACT) and you are confident that there are duplicate transactions
in the trail that could cause Replicat to abend. This option requires the use
of a checkpoint table. If the database is Oracle, this option is valid only
for Replicat in nonintegrated mode. In case of Integrated mode and automatic
target trail file regeneration, the Integrated mode handles the duplicate
```

transactions transparently. The default is FILTERDUPTRANSACTIONS.

THREADS thread-list
Valid for SKIPTRANSACTION, ATCSN, and AFTERCSN when Replicat is in coordinated mode. Not valid for START REPLICAT without those options. Starts the specified Replicat thread or threads at the specified location.

    thread-list

    A comma-delimted list of ranges in the format of threadIDlow-threadIDhigh, threadIDlow-threadIDhigh.

Examples

Example 1

START REPLICAT finance

Example 2

The following starts Replicat at an Oracle-specific CSN.

START REPLICAT finance, ATCSN 6488359

Example 3

The following causes threads 4 and 5 of a coordinated Replicat to skip the first transaction after their last checkpoint when Replicat is started. If this were a 10-thread coordinated Replicat, threads 0-3 and 6-10 would all start at the normal start point, that of their last checkpoint.

START REPLICAT fin SKIPTRANSACTION THREADS(4-5)

Example 4

The following example causes threads 1-3 of a coordinated Replicat to start at CSN 6488359, threads 9-10 to start after CSN 6488360, and threads 7 and 8 to skip the first transaction after its last checkpoint.

START REPLICAT fin ATCSN 6488359 THREADS(1-3), AFTERCSN 6488360 THREADS(9-10), SKIPTRANSACTION THREADS(7,8)

START SERVICE

Use START SERVICE to start the specified Oracle GoldenGate services.

service-name-wildcard

The name of an service or a wildcard (*) to specify multiple services. Valid services are ADMINSRVR, DISTSRVR, RECVSRVR, and PMSRVR.

Example

START SERVICE ADMIN*

STATS DISTPATH

```
Use STATS DISTPATH  to get the statistics for a distribution path.

path-name


The name of the distribution path.


-----------------------------------------------
STATS ER

Use STATS ER to get the processing statistics for the specified wildcarded
groups.

group-name-wildcard

The name of a group or a wildcard (*) to specify multiple groups. For
example, T* starts all groups whose names begin with T.


-----------------------------------------------
STATS EXTRACT

Use STATS EXTRACT to display statistics for one or more Extract groups. The
output
includes DML and DDL operations that are included in the Oracle GoldenGate
configuration.

To get the most accurate number of operations per second that are being
processed,
do the following.

1. Issue the STATS EXTRACT command with the RESET option.

2. Issue the STATS EXTRACT REPORTRATE command. The LATEST STATISTICS field shows
the operations per second.

group-name-wildcard

The name of an Extract group or a wildcard (*) to specify multiple groups. For
example, T* returns statistics for all Extract groups whose names start with T.

TOTAL

Displays totals since process startup.

    DAILY

    Displays totals since the start of the current day.

    HOURLY

    Displays totals since the start of the current hour.

    LATEST

    Displays totals since the last RESET command.

    RESET
    Resets the counters in the LATESTstatistical field.
```

```
TABLE table-name

Displays statistics only for the specified table or a group of tables
specified with a wildcard (*). The table name or wildcard specification
must be fully qualified with the two-part or three-part name, for example
hr.empor *.*.*.

TOTALSONLY table-name

Summarizes the statistics for the specified table or a group of tables
specified with a wildcard (*). The table name or wildcard specification
must be fully qualified with the two-part or three-part name, for example
hr.empor *.*.*.

REPORTCDR

Shows statistics for Conflict Detection and Resolution. Statistics include:

* Total CDR conflicts

* CDR resolutions succeeded

* CDR resolutions failed

* CDR INSERTROWEXISTS conflicts

* CDR UPDATEROWEXISTS conflicts

* CDR DELROWEXISTS conflicts

* CDR DELROWMISSING conflicts

REPORTCHARCONV

Use only when TABLE parameters have a TARGET clause and character-set
conversion is performed. The following statistics are added to the STATS
output:

Total column character set conversion failure: the number of validation
or conversion failures in the current Extract run.

Total column data truncation: the number of times that column data was
truncated in the current Extract run as the result of character set conversion

REPORTFETCH | NOREPORTFETCH

Controls whether or not statistics about fetch operations are included
in the output. The default is NOREPORTFETCH. See STATOPTIONS for defaults
that control fetching and options for altering fetch behavior. The output of
REPORTFETCH is as follows:

* row fetch attempts: The number of times Extract attempted to fetch a column
value from the database when it could not obtain the value from the transaction
log.

* fetch failed: The number of row fetch attemptsthat failed.

* row fetch by key: Valid for Oracle. The number of row fetch attempts that
were made by using the primary key. The default is to fetch by row ID.
```

```
REPORTRATE time-units

Displays statistics in terms of processing rate rather than absolute values.

Valid values:

HR
MIN
SEC

Examples

Example 1

The following example displays total and hourly statistics per minute for a
specific table, and it also resets the latest statistics and outputs fetch
statistics.
STATS EXTRACT finance, TOTAL, HOURLY, TABLE hr.acct, REPORTRATE MIN, RESET,
REPORTFETCH

Example 2

STATS EXTRACT ext, LATEST, REPORTFETCH


-----------------------------------------------
STATS REPLICAT

Use STATS REPLICAT to display statistics for one or more Replicat groups.
Thread statistics for a coordinated Replicat group are provided as follows.

Thread Lag Gap

The difference between the maximum lag and the minimum lag among all threads.

Coordinated Total DDLs
The total number of coordinated DDL transactions.

Coordinated Total PK-Update Transactions
The total number of coordinated transactions that involved an update to a
primary key.

Coordinated Total EMI Transactions
The total number of coordinated EVENTACTIONS events.

Total Transactions with User-requested Coordination
The total number of coordinations that were explicitly requested in the
configuration by means of the COORDINATED option of the MAP parameter.

Average Coordination Time
The average time (in seconds) spent in coordination among all threads.

group-name-wildcard

The name of a Replicat group or a wildcard (*) to specify multiple groups.
For example, T* shows statistics for all Replicat groups whose names begin with
T.


TOTAL
```

```
Displays totals since process startup.

    DAILY

    Displays totals since the start of the current day.

    HOURLY

    Displays totals since the start of the current hour.

    LATEST

    Displays totals since the last RESET command.

    RESET

    Resets the counters in the LATEST statistical field.

TABLE table-name

Displays statistics only for the specified table or a group of tables
specified with a wildcard (*).

TOTALSONLY table-name

Summarizes the statistics for the specified table or a group of tables
specified with a wildcard (*).

REPORTCDR

Shows statistics for Conflict Detection and Resolution. Statistics include:

* Total CDR conflicts

* CDR resolutions succeeded

* CDR resolutions failed

* CDR INSERTROWEXISTS conflicts

* CDR UPDATEROWEXISTS conflicts

* CDR DELROWEXISTS conflicts

* CDR DELROWMISSING conflicts

REPORTCHARCONV

Reports statistics for character validation when character-set conversion
is performed. The following statistics are added to the STATS output:

Total column character set conversion failure: the number of validation
or conversion failures in the current Replicat run.

Total column data truncation: the number of times that column data was
truncated in the current Replicat run as the result of character set conversion.

REPORTDETAIL | NOREPORTDETAIL

Controls whether or not the output includes operations that were not replicated
as the result of collision errors. These operations are reported in the
```

regular statistics (inserts, updates, and deletes performed) plus as statistics in the detail display, if enabled. For example, if 10 records were insert operations and they were all ignored due to duplicate keys, the report would indicate that there were 10 inserts and also 10 discards due to collisions. The default is REPORTDETAIL. See 刀ATOPTIONS刀

REPORTRATE time-units

Displays statistics in terms of processing rate rather than absolute values.

HR
Sets the processing rate in terms of hours.

MIN
Sets the processing rate in terms of minutes.

SEC
Sets the processing rate in terms of seconds.

Examples

Example 1

The following example displays total and hourly statistics per minute for a specific
table, and it also resets the latest statistics. Statistics for discarded operations
are not reported.

```
STATS REPLICAT finance, TOTAL, HOURLY, TABLE sales.acct, REPORTRATE MIN, RESET,
NOREPORTDETAIL
```

Example 2

The following example displays the same statistics as the previous example, but for thread 3 of a coordinated Replicat group.

```
STATS REPLICAT fin003, TOTAL, HOURLY, TABLE sales.acct, REPORTRATE MIN, RESET,
NOREPORTDETAIL
```

STATUS DEPLOYMENT

Use STATUS DEPLOYMENT to see the status of the specified Oracle GoldenGate deployments.

deployment-name-wildcard

The name of a deployment or a wildcard (*) to specify multiple deployments. For example, T* displays the status of all deployments whose names begin with T.

STATUS ER

Use STATUS ER to get the state of the specified wildcarded Extract or Replicat groups.

```
STATUS ER group-name-wildcard
```

```
group-name-wildcard

The name of a  group or a wildcard (*) to specify multiple groups.
For example, T* shows statistics for all groups whose names begin with T.


------------------------------------------------
STATUS EXTRACT

Use STATUS EXTRACT to determine whether or not an Extract is running. A status
of RUNNING can mean one of the following:

* Active: Running and processing (or able to process) data. This is the
normal state of a process after it is started.

* Suspended: The process is running, but suspended due to an EVENTACTIONS SUSPEND
action. In a suspended state, the process is not active, and no data can be
processed, but the state of the current run is preserved and can be continued by
issuing the RESUME command in Admin Client. The RBA in the INFO command reflects
the last
checkpointed position before the suspend action. To determine whether the state
is active or suspended, issue the SEND EXTRACT command with the STATUS option.

STATUS EXTRACT group-name-wildcard
[ TASKS | ALLPROCESSES ]

group-name-wildcard

The name of an Extract group or a wildcard (*) to specify multiple groups.
For example, T* returns status for all Extract groups whose names begin with T.

TASKS

Displays status only for Extract tasks. By default, tasks are not displayed
unless you specify a single Extract group (without wildcards).

ALLPROCESSES

Displays status for all Extract groups, including tasks.

Examples

Example 1

STATUS EXTRACT finance

Example 2

STATUS EXTRACT fin*


------------------------------------------------
STATUS REPLICAT

Use STATUS REPLICAT to determine whether or not Replicat is running. There are
the following four possible statuses:

Abended

The process has abnormally ended.
```

```
Running

Means one of the following:

* Active: Running and processing (or able to process) data. This is the normal
state of a process after it is started.

* Suspended: The process is running, but suspended due to an EVENTACTIONS SUSPEND
action. In a suspended state, the process is not active, and no data can be
processed, but the state of the current run is preserved and can be continued
by issuing the RESUME command in Admin Client. The RBA in the INFO command
reflects the
last checkpointed position before the suspend action. To determine whether the
state is active or suspended, issue a SEND EXTRACT|REPLICAT group_name STATUS
command. For more information, see SEND EXTRACT or SEND REPLICAT.

Starting

The process is starting.

Stopped

The process was stopped.

group-name-wildcard

The name of a Replicat group or a wildcard (*) to specify multiple
groups. For example, T* shows status for all Replicat groups whose names
begin with T.

TASKS

Displays status only for Replicat tasks. By default, tasks are not displayed
unless you specify a single Replicat group (without wildcards).

ALLPROCESSES

Displays status for all Replicat groups, including tasks.

Examples

Example 1

STATUS REPLICAT finance

Example 2

STATUS REPLICAT fin*

-----------------------------------------------
STATUS SERVICE

Use STATUS SERVICE to display status of the specified Oracle GoldenGate services.

service-name-wildcard

The name of an service or a wildcard (*) to specify multiple services. For
example, T* statuses all services whose names begin with T.


-----------------------------------------------
```

```
STOP DEPLOYMENT

Use STOP DEPLOYMENT to stop one or more deployments.

deployment-name-wildcard

The name of the deployment or a wildcard (*) to specify multiple deployments.
For
example, P* stops all services whose names begin with P.

Example


STOP DEPLOYMENT Phoenix


------------------------------------------------
STOP DISTPATH

Use STOP DISTPATH to stop a distribution path.

path-name

The name of the distribution path.


------------------------------------------------
STOP ER

Use STOP ER to stop the specified wildcarded groups.

group-name-wildcard

The name of a Extract or Replicat group or a wildcard (*) to specify multiple
groups. For example, T* shows status for all groups whose names
begin with T.

!

(Exclamation point) Bypasses the prompt that confirms intent to stop the groups.


------------------------------------------------
STOP EXTRACT

Use STOP EXTRACT to stop Extract gracefully. The command preserves the state
of synchronization for the next time Extract starts.

If there are open, long-running transactions when you issue STOP EXTRACT, you
might be advised of the oldest transaction log file that will be needed for
that transaction when Extract is restarted. You can use the SEND EXTRACT option
of SHOWTRANS to view details and data of those transactions and then, if
desired,
use the SKIPTRANS or FORCETRANS options to skip the transaction or force it to
be
written as a committed transaction to the trail. See SEND EXTRACT.

group-name-wildcard

The name of an Extract group or a wildcard (*) to specify multiple groups.
For example, T* stops all Extract processes for groups whose names begin with T.
```

!

(Exclamation point) Bypasses the prompt that confirms intent to stop the Extract.

Example

```
STOP EXTRACT finance
```

STOP REPLICAT

Use STOP REPLICAT to stop Replicat cleanly. This command preserves the state of synchronization for the next time Replicat starts.

In a clean shutdown of a coordinated Replicat, the coordinator thread attempts to stop all of the threads on the same transaction boundary. If the shutdown of a coordinated Replicat is not clean, the threads may stop at different positions in the trail file. If this happens, START REPLICAT writes a warning if the parameter file was changed since the prior run and raises an error if the number of threads was changed. To resolve these problems and start Replicat again, see Administering Oracle GoldenGate.

group-name-wildcard

The name of a Replicat group or a wildcard (*) to specify multiple groups. For example, T* stops all Replicat groups whose names begin with T.

!
(Exclamation point) Stops Replicat immediately. The transaction is stopped and the process terminates.

Example

```
STOP REPLICAT finance
```

STOP SERVICE

Use this to stop the specified Oracle GoldenGate services.

service-name-wildcard

The name of an service or a wildcard (*) to specify multiple services. For example, T* stops all services whose names begin with T.

SYNCHRONIZE REPLICAT

Use SYNCHRONIZE REPLICAT to return all of the threads of a coordinated Replicat to the same position in the trail file after an unclean shutdown. This position is the maximum checkpoint position of all of the threads, in other words, the most recent trail record processed among all of the threads. When SYNCHRONIZE REPLICAT is issued, all threads are started and allowed to process transactions until they reach the maximum checkpoint position, and then Replicat stops.

For more information about how to use SYNCHRONIZE REPLICAT to recover a coordinated Replicat after an unclean shutdown, or to enable repartitioning

of data among different threads, see Administering Oracle GoldenGate.

group-name-wildcard

The name of a Replicat group or a wildcard (*) to specify multiple groups. For example, T* synchronizes the threads of all Replicat groups whose names begin with T. The threads synchronize to the same position within their group, not to the same position across all Replicat groups being synchronized with this command.

Example

```
SYNCHRONIZE REPLICAT repA
```

UNDELETE MASTERKEY VERSION

Use the UNDELETE MASTERKEY VERSION command to remove the deletion mark from a master key version, thus retaining that version if the PURGE WALLET command is used. Only one version can be unmarked per UNDELETE MASTERKEY command. See DELETE MASTERKEY to mark a version of a master key for deletion.

version-number

The version that is to be unmarked for deletion.

Example

This command unmarks version 3 of the master key and returns a message similar to the one shown.

```
UNDELETE MASTERKEY VERSION 3
Version 3 of Masterkey 'OGG_DEFAULT_MASTERKEY' undeleted from wallet
at location './ wallet'.
```

UNREGISTER EXTRACT

Use UNREGISTER EXTRACT to remove the registration of an Extract group from an Oracle Database. UNREGISTER EXTRACT is valid only for a primary Extract group.

To register an Extract group with the database, use the REGISTER EXTRACT command.

To upgrade an Extract from classic capture mode to integrated capture mode, use the ALTER EXTRACT command.

group-name

The name of the Extract group that is to be unregistered from the database. Do not use a wildcard. This group must currently be registered with the database.

LOGRETENTION

Disables log retention for the specified Extract group and removes the underlying

Oracle Streams capture process. Use UNREGISTER EXTRACT with LOGRETENTION only if you no longer want to capture changes with this Extract group. The log-retention feature is controlled with the LOGRETENTION option of the TRANLOGOPTIONS

parameter.

Before using UNREGISTER EXTRACT with LOGRETENTION, stop Extract with the STOP EXTRACT
command. Next, issue the DBLOGIN command with the privileges shown in Examples 1-2.

DATABASE

Disables integrated capture mode for the Extract group.

This command removes the database capture (mining) server that has the same name as the Extract group. For additional information about support for, and configuration of, the Extract capture modes, see Using Oracle GoldenGate for Oracle Database.

Before using UNREGISTER EXTRACT with DATABASE, do the following:

1. Stop Extract with the STOP EXTRACT command.

2. Log in to the mining database with the DBLOGIN or MININGDBLOGIN command with the
privileges granted in the dbms_goldengate_auth.grant_admin_privilege procedure. For local capture, DBLOGINis required. For downstream capture, DBLOGIN and MININGDBLOGIN are both required.

3. Delete the Extract group with DELETE EXTRACT.

Examples

Example 1

UNREGISTER EXTRACT sales LOGRETENTION

Example 2

UNREGISTER EXTRACT sales DATABASE

UNREGISTER REPLICAT

Use the UNREGISTER REPLICAT command to unregister a Replicat group from a target Oracle Database to disable integrated Replicat mode. Use this command only if you forcibly deleted the Replicat group. UNREGISTER REPLICAT should not be used when deleting Replicat in the normal manner, where you first stop Replicat and then issue the DELETE REPLICAT command.

Before issuing this command, issue the DBLOGIN command as the Replicat database user with privileges granted through dbms_goldengate_auth.grant_admin_privilege.

group-name-wildcard

The name of a Replicat group or a wildcard (*) to specify multiple groups. For example, T* unregisters all Replicat groups whose names begin with T.

DATABASE

Required keyword to unregister from the target database. Removes the database inbound server that is associated with this Replicat.

```
Example

UNREGISTER REPLICAT sales DATABASE

-----------------------------------------------
UPGRADE CHECKPOINTTABLE

Not valid for Replicat for Java, Oracle GoldenGate Applications Adapter, or
Oracle
GoldenGate for Big Data.

Use the UPGRADE CHECKPOINTTABLE command to add a supplemental checkpoint table
when
upgrading Oracle GoldenGate.

table-name

The name of the checkpoint table. The table name is required and must include
the schema
name.

Example

UPGRADE CHECKPOINTTABLE ggs.fin_check

-----------------------------------------------
VERSIONS

Use VERSIONS to display operating system and database version information. For
ODBC connections, the driver version is also displayed. To include database
information in the output, issue a DBLOGIN command before issuing VERSIONS to
establish a database connection.


-----------------------------------------------
VIEW DISCARD

Use VIEW DISCARD to display the discard file that is generated by Extract or
Replicat.
The SET PAGER value is used to determine pagination of the output.

report-name

The name of the report to display the discard file. For EXTRACT "EXX", these
report
names are valid:

  - EXX
  - EXX0
  - EXX1
  - ...
  - EXX9

No other values are valid

-----------------------------------------------
VIEW ENCKEYS

Use VIEW ENCKEYS to display the  contents of the ENCKEYS file in
read-only mode on-screen.
```

```
-----------------------------------------------
VIEW GLOBALS

Use VIEW GLOBALS to display the contents of the GLOBALS parameter file in read-
only
mode on-screen. The SET PAGER value is used to determine pagination of the
output.


-----------------------------------------------
VIEW MESSAGES

Use VIEW MESSAGES to display the Oracle GoldenGate message log (ggserr.log
file). The
SET PAGER value is used to determine pagination of the output.


-----------------------------------------------
VIEW PARAMS

Use VIEW PARAMS to view the contents of a parameter file. The SET PAGER value
is used to determine pagination of the output.

file-name

Shows the specified file.

Example

VIEW PARAMS finance


-----------------------------------------------
VIEW REPORT


Use VIEW REPORT to view the process report that is Generated by Extract or
Replicat.
Each process generates a new report and discard file upon startup. The SET PAGER
value is used to determine pagination of the output.

Reports and discard files are aged whenever a process starts. Old files are
appended with a sequence number, for example finance0.rpt, finance1.rpt, and
so forth, or discard0.dsc, discard1.dsc, and so forth. To view old files, use
the [n] option. To view the current report or discard file, use the command
without the [n] option.

report-name

For EXTRACT "EXX", these report names are valid:

  - EXX
  - EXX0
  - EXX1
  - ...
  - EXX9

No other values are valid
```

Example

The following displays an old report file (number 3) for the ordersgroup.

```
VIEW REPORT orders3


-----------------------------------------------
```

Part III
Oracle GoldenGate Classic Architecture

You can use the Oracle GoldenGate Classic Architecture to configure and manage your data replications from the command line.

This part describes the commands, parameters, and options that you can use with the Oracle GoldenGate Classic Architecture.

Topics:

- Collector Parameters
- Manager Parameters
- Oracle GoldenGate Native Commands
- Oracle GoldenGate GGSCI Commands
  You can use the Oracle GoldenGate Software Command Interface (GGSCI) commands to create data replications. This is the command interface between you and Oracle GoldenGate functional components.

6
Collector Parameters

This chapter describes the parameters for the Collector process and includes the following topics:
The Collector process operates on the target system to receive incoming data and write it to the trail.

Topics:

- Overview of the Collector Process
- Summary of Collector Parameters
- -B
- -b
- -cp
- -d
- -E
- -e
- -ENCRYPT
- -f
- -g
- -h
- -k
- -KEYNAME
- -l
- -m
- -P
- -p
- -R
- -x

6.1 Overview of the Collector Process

Typically, Oracle GoldenGate users do not interact with the Collector process. This is known as a *dynamic collector*. It is started dynamically by the Manager process, but parameters may be sent to Collector as options of certain Extract or Replicat parameters.

As an alternative to allowing Manager to run Collector, you can run a *static* Collector manually by running the SERVER program at the command line with the following syntax and input parameters as shown:

```
server parameter [parameter] [...]
```

Collector parameters are case-sensitive and must be preceded by a dash. For example, `-e` and `-E` are two different parameters, with entirely different results.

6.2 Summary of Collector Parameters

The following is a summary of available Collector parameters.

Table 6-1 Collector Parameters

Parameter	Description
-B	Directs the Collector to buffer files.
-b	Specifies the default the file buffer size.
-cp	Specifies the name of the checkpoint file that Collector maintains for an alias Extract group.
-d	Specifies the name of a local definitions file that was generated by the DEFGEN utility.
-E	Converts incoming header and data to EBCDIC format from ASCII.
-e	Directs Collector to respond to specific formatting error conditions in custom ways.
-ENCRYPT	Specifies the type of encryption being passed from the Extract process, as specified with the `RMTHOST` parameter in the Extract parameter file.
-f	Forces all file writes to be flushed to disk before returning a success status to the Extract process.
-g	Supports files that are larger than 2GB (Solaris only).
-h	Specifies the name or IP address of the source system.
-k	Directs Collector to terminate when the Extract process that it is serving disconnects.
-KEYNAME	Specifies the name of a key that is defined in the local `ENCKEYS` lookup file.
-l	Logs output to the specified file.
-m	Specifies the Manager port
-P	Specifies a local file that contains Collector parameters.
-p	Specifies a TCP/IP port number.
-R	Replaces invalid numeric ASCII data with an alternate value.
-x	Specifies a discard file to store records that could not be processed by Oracle GoldenGate.

6.3 -B

Specifies the default file buffer size.

Syntax

```
-Bsize
```

There is no space between `-B` and the *`size`* value.

If *`size`* is not specified, then the default size is your C library default file buffer size (`BUFSIZ`). The minimum value is 16384 (16K) and the max is 16777216 (16MB).

Example

```
-B16384
```

6.4 -b

Specifies the default file buffer size.

Syntax:

`-B` *`size`*

If *`size`* is not specified, then the default size is your C library default file buffer size (`BUFSIZ`). The minimum value is 16 KB and the maximum is 16 MB; zero (0) indicates unbuffered.

6.5 -cp

Specifies the name of the checkpoint file that Collector maintains for an alias Extract group. The checkpoint file is used to determine whether the passive Extract is running or not. It is running when the checkpoint file is locked by Collector (shown as the `server` program in the Oracle GoldenGate installation directory).

`-cp` must be used with the `-h` and `-p` parameters.

Syntax

`-cp` *`checkpoint_file`*

`cp`
Must be lower case.

`checkpoint_file`
The name of the file to which the passive Extract group writes its checkpoints. The name of the passive Extract group and the name of the checkpoint file are identical.

6.6 -d

Specifies the name of a local definitions file that was generated by the `DEFGEN` utility. The file contains the definitions of tables that reside on a remote system. For more information about definitions files, see *Administering Oracle GoldenGate*.

Syntax

`-d` *`definitions_file`*

`d`
Must be lower case.

definitions_file
The name of the definitions file, exactly as specified when DEFGEN was run.

6.7 -E

Converts incoming header and data to EBCDIC format from ASCII. By default, Oracle GoldenGate does not convert the data.

Syntax

```
-E
```

E
Must be upper case.

6.8 -e

Directs Collector to respond to specific formatting error conditions in custom ways. Default values are almost always sufficient. To specify more than one error type, use `-e` multiple times. For example:

```
-e OLD CONTINUE -e NEW DISCARD.
```

Syntax

```
-e type action
```

e
Must be lower case.

type
Specifies the type of error that generates the response and 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 (that is, DEFGEN must 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 than the table's current definition, which is considered a normal condition. The default action is `CONTINUE`.

OUTOFSYNC
Checks for records that cannot be converted according to the definition provided. The default action is `ABEND`.

action
Specifies the response to the error and 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 `-x`). Collector sends a warning to the error file for the first discarded record and continues to process records.

6.9 -ENCRYPT

Specifies the type of encryption being passed from the Extract process, as specified with the `RMTHOST` parameter in the Extract parameter file.

Syntax

```
-ENCRYPT {NONE | BLOWFISH}
```

ENCRYPT
Must be upper case.

NONE
Specifies that the data will not be encrypted.

BLOWFISH
Specifies `BLOWFISH` encryption. If using `BLOWFISH`, also specify the `-KEYNAME` option.

6.10 -f

Forces all file writes to be flushed to disk before returning a success status to the Extract process. By default, the file system buffers the I/O because it is more efficient than flushing to disk with every operation. Generally, the performance benefits outweigh the small risk that data could be lost if the system fails after an I/O is confirmed successful, but before the buffer actually is flushed to disk. Use `-f` if this risk is unacceptable, with the understanding that it can compromise the performance of Oracle GoldenGate

Syntax

```
-f
```

f
Must be lower case.

6.11 -g

Supports files that are larger than 2GB (Solaris only).

Syntax

```
-g
```

g
Must be lower case.

6.12 -h

Specifies the name or IP address of the source system. Use this option when using an alias Extract on the target that is associated with an Extract running in PASSIVE mode on the source. It causes Collector to operate in connection mode. In this mode, it initiates a TCP/IP connection to the source Extract, instead of waiting for a connection request from Extract. Must be used with the -p Collector option.

Syntax

```
-h {host_name | IP_address}
```

h
Must be lower case.

host_name
Specifies the source system by its DNS host name.

IP_address
Specifies the source system by its IP address.

6.13 -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.

Syntax

```
-k
```

k
Must be lower case.

6.14 -KEYNAME

Specifies the name of a key that is defined in the local ENCKEYS lookup file. Use if BLOWFISH is specified for -ENCRYPT.

Syntax

```
-KEYNAME key_name
```

KEYNAME
Must be upper case.

key_name
The name of the key as it appears in the ENCKEYS file.

6.15 -l

Logs output to the specified file.

Syntax

```
-l file_name
```

l
Must be lower case.

file_name
The fully qualified name of the output file.

6.16 -m

Specifies the Manager port number.

Syntax

```
-m number
```

m
Must be lower case.

number
The Manager port number.

6.17 -P

Specifies a local file that contains Collector parameters. Parameters in this file override parameters sent from the Extract process.

Syntax

```
-P file_name
```

P
Must be upper case.

file_name
The fully qualified name of the parameter file.

6.18 -p

Specifies a TCP/IP port number as follows:

- For a regular Extract or regular data pump: the port on which the Collector process listens for connection requests from Extract.
- For an Extract or data pump running in `PASSIVE` mode: the port on which Extract or the data pump listens for connection requests from Collector. Must be used with the `-h host` parameter in this case.

Syntax

```
-p port
```

p
Must be lower case.

port
The port number. The default is port 7819.

6.19 -R

Replaces invalid numeric ASCII data with an alternate value.

Syntax

```
-R value
```

R
Must be upper case.

value
The replacement value. The default is to replace with 0. Specify one of the following alternate values:

number
Replaces invalid data with the specified number.

NULL
Replaces invalid data with `NULL` if the target column accepts `NULL` values. Otherwise, replaces with `0`.

UNPRINTABLE
Rejects any column with unprintable data. The process stops and reports the bad value.

NONE
Does not replace numeric data. Oracle GoldenGate attempts to replicate the data as-is.

6.20 -x

Specifies a discard file to store records that could not be processed by Oracle GoldenGate.

Syntax

```
-x discard_file
```

x
Must be lower case.

discard_file
The fully qualified name of the discard file.

7

Manager Parameters

This chapter describes the parameters you can use to control the Manager process. Manager is the control process of Oracle GoldenGate, To know more, see Configuring Manager and Network Communications in*Administering Oracle GoldenGate*.

Topics:

- Summary of Manager Parameters
- ACCESSRULE
- AUTORESTART
- AUTOSTART
- BOOTDELAYMINUTES
- CHARSET
- CHECKMINUTES
- COMMENT | --
- DOWNREPORT
- DYNAMICPORTLIST
- MONITORING_HEARTBEAT_TIMEOUT
- LAGCRITICAL
- LAGINFO
- LAGREPORT
- PURGEMARKERHISTORY
- PORT
- SOURCEDB
- PURGEDDLHISTORY | PURGEDDLHISTORYALT
- PURGEOLDEXTRACTS for Extract and Replicat
- PURGEOLDEXTRACTS for Manager
- PURGEOLDTASKS
- STARTUPVALIDATIONDELAY[CSECS]
- USERIDALIAS
- VERIDATAREPORTAGE

7.1 Summary of Manager Parameters

Manager is the parent process of Oracle GoldenGate and is responsible for the management of its processes, resources, user interface, and the reporting of thresholds and errors. In most cases default settings for Manager suffice.

Parameter	Description
ACCESSRULE	Adds security access rules for Manager.
CHARSET	Specifies a multibyte character set for the process to use instead of the operating system default when reading the parameter file.
COMMENT | --	Allows insertion of comments in a parameter file.
SOURCEDB	Specifies a data source name as part of the login information.
USERIDALIAS	Provides login information for Manager when it needs to access the database.

Manager parameters: port management.

Parameter	Description
DYNAMICPORTLIST	Specifies the ports that Collector can dynamically allocate.
PORT	Establishes the TCP/IP port number on which Manager listens for requests.

Manager parameters: Process management.

Parameter	Description
AUTORESTART	Specifies processes to be restarted by Manager after a failure.
AUTOSTART	Specifies processes to be started when Manager starts.
BOOTDELAYMINUTES	Determines how long after system boot time Manager delays until performing main processing activities. This parameter supports Windows.
MONITORING_HEARTBEAT_TIMEOUT	Sets a process as non-responsive in a specified number of seconds.
UPREPORT	Determines how often process heartbeat messages are reported.

Manager parameters: Event management.

Parameter	Description
DOWNREPORT	Controls the frequency for reporting stopped processes.
LAGCRITICAL	Specifies a lag threshold that is considered critical and generates a warning to the error log.
LAGINFO	Specifies a lag threshold at which an informational message is reported to the error log.
LAGREPORT	Sets an interval for reporting lag time to the error log.

Manager parameters: Maintenance.

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

Parameter	Description
PURGEDDLHISTORY | PURGEDDLHISTORYALT	Purges rows from the Oracle DDL history table when they are no longer needed.
PURGEMARKERHISTORY	Purges Oracle marker table rows that are no longer needed.
PURGEOLDEXTRACTS for Extract and Replicat	Purges trail data that is no longer needed.
PURGEOLDEXTRACTS for Manager	Purges trail files when processing has finished; Oracle recommends use of this parameter rather than the PURGEOLDEXTRACTS for Extract and Replicat parameter.
PURGEOLDTASKS	Purges Extract and Replicat tasks after a specified period of time.
STARTUPVALIDATIONDELAY[CSECS]	Sets a delay time after which Manager checks that processes are still running after startup.
VERIDATAREPORTAGE	Purges old Veridata report files when they have reached the specified age limit.

7.2 ACCESSRULE

Valid for Manager

Use ACCESSRULE to control connection access to the Manager process and the processes under its control. You can establish multiple rules by specifying multiple ACCESSRULE statements in the parameter file and control their priority. There is no limit to the number of rules that you can specify. To establish priority, you can either list the rules in order from most important to least important, or you can explicitly set the priority of each rule with the PRI option.

Default

None

Syntax

```
ACCESSRULE[, PROG program_name][, IPADDR address][, PRI rule][, login_ID]{,
ALLOW | DENY}
```

Argument	Description
PROG *program_name*	Specifies connection security for a specific Oracle GoldenGate program or multiple programs specified with a wildcard. If one of these options is not specified, the access rule applies to all programs that Manager starts, stops, or kills. Valid values: • GGSCI: Secures access to the GGSCI command-line interface. • GUI: Secures access to Oracle GoldenGate from the Activity Console. • MGR | MANAGER: Secures access to all inter-process commands controlled by Manager, such as START, STOP, and KILL • REPLICAT: Secures connection to the Replicat process. • COLLECTOR | SERVER: Secures the ability to dynamically create a Collector process. • * (asterisk): Wildcard. Use a wildcard to specify all of the preceding options.
IPADDR *address*	Permits access to Manager from the host with the specified IP address.
PRI *rule*	Specifies a priority for each ACCESSRULE statement. Valid values are from 1 through 99, with 1 being the highest priority and 99 being the lowest. Rules that have priorities assigned can appear in any order in the parameter file.
login_ID	Permits access based on a user password. This option requires specifying USER and PASSWORD options with the RMTHOST parameter. The syntax for *login_ID* is: USER *user*, PASSWORD *password*, [ENCRYPTKEY *keyname*] Valid values: • *user* : The user specified with the USER option of the RMTHOST parameter. • *password*: The password specified with the PASSWORD option of the RMTHOST parameter. • *keyname*: Optional. Specifies an encryption key in the ENCKEYS file. When ENCRYPTKEY *keyname* is used as part of the login ID, Oracle GoldenGate looks up the key in the ENCKEYS file on the target system and uses it to decrypt the corresponding password. If the decrypted password matches the password supplied with the *password* portion of the login ID option, the rule passes.
ALLOW | DENY	Determines whether the rule specified with ACCESSRULE permits or denies access. Either ALLOW or DENY is required.

Example 1

The following access rules allow any nodes that begin with IP address 205 or the node 194.168.11.102 to access the requested services. All others are denied.

```
ACCESSRULE, PROG *, IPADDR 194.168.11.102, ALLOW ACCESSRULE, PROG *, IPADDR
205.*, ALLOW ACCESSRULE, PROG *, IPADDR *, DENY
```

Example 2

The following access rules have been assigned explicit priority levels through the PRI option. These rules allow any user to access the Collector process (the SERVER program), and in addition, allow the IP address 122.11.12.13 to access GGSCI commands. Access to all other Oracle GoldenGate programs is denied.

```
ACCESSRULE, PROG *, DENY, PRI 99ACCESSRULE, PROG SERVER, ALLOW, PRI 1ACCESSRULE,
PROG GGSCI, IPADDR 122.11.12.13, PRI 1
```

Example 3

The following access rules are the same as Example 2, but they assign priority by means of their order in the parameter file, instead of the `PRI` option.

```
ACCESSRULE, PROG SERVER, ALLOWACCESSRULE, PROG GGSCI, IPADDR
122.11.12.13ACCESSRULE, PROG *, DENY
```

Example 4

The following access rule grants access to all programs to the user `JOHN`.

```
ACCESSRULE, PROG *, USER JOHN, PASSWORD OCEAN1
```

Example 5

The following access rule grants access to all programs to the user JOHN and designates an encryption key to decrypt the password. If the password provided with `PASSWORD` matches the one in the `ENCKEYS` lookup file, connection is granted.

```
ACCESSRULE, PROG *, USER JOHN, PASSWORD OCEAN1, ENCRYPTKEY lookup1
```

7.3 AUTORESTART

Valid For

Manager

Description

Use the `AUTORESTART` parameter to start one or more Extract and Replicat processes automatically after they fail. `AUTORESTART` provides fault tolerance when something temporary interferes with a process, such as intermittent network outages or programs that interrupt access to transaction logs.

You can use multiple `AUTORESTART` statements in the same parameter file.

To apply this parameter to an Extract group that is created in `PASSIVE` mode, use it for the Manager that is on the target system where the associated alias Extract group resides. Oracle GoldenGate will send the start command to the source system. If `AUTORESTART` is used locally for a passive Extract group, it is ignored.

If Manager encounters an out-of-order transaction upon restart, it will not restart Extract. Instead, it will log a warning that notifies you to use the `ETROLLOVER` option of `SEND EXTRACT` to advance the trail to skip the transaction that caused the error.

Default

Do not auto-restart

Syntax

```
AUTORESTART {EXTRACT | REPLICAT | ER} group_name
[, RETRIES number]
[, WAITMINUTES minutes]
[, WAITSECONDS seconds]
[, RESETMINUTES minutes]
[, RESETSECONDS seconds]
```

EXTRACT
Restarts Extract automatically.

REPLICAT
Restarts Replicat automatically.

ER
Restarts Extract and Replicat automatically.

group_name
A group name or wildcard specification for multiple groups. When wildcarding is used, Oracle GoldenGate starts all groups of the specified process type on the local system that satisfy the wildcard, except those in `PASSIVE` mode.

RETRIES *number*
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 | WAITSECONDS *{minutes | seconds}*
The amount of time, in minutes or seconds, 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 2 minutes or 120 seconds.

RESETMINUTES | RESETSECONDS *{minutes | seconds}*
The window of time, in minutes or seconds, during which retries are counted. The default is 120 minutes (2 hours) or 7200 seconds. After the time expires, the number of retries reverts to zero.

Example

In the following example, Manager tries to start all Extract processes three times after failure within a one hour time period, and it waits five minutes before each attempt.

```
AUTORESTART EXTRACT *, RETRIES 3, WAITMINUTES 5, RESETMINUTES 60
```

7.4 AUTOSTART

Valid For

Manager

Description

Use the AUTOSTART parameter to start one or more Extract and Replicat processes automatically when Manager starts. AUTOSTART ensures that no process groups are overlooked and that synchronization activities start immediately.

You can use multiple AUTOSTART statements in the same parameter file.

To apply this parameter to an Extract group that is created in PASSIVE mode, use it for the Manager that is on the target system where the associated alias Extract group resides. Oracle GoldenGate will send the start command to the source system. If AUTOSTART is used locally for a passive Extract group, it is ignored.

If Manager encounters an out-of-order transaction upon restart, it will not restart Extract. Instead, it will log a warning that notifies you to use the ETROLLOVER option of SEND EXTRACT to advance the trail to skip the transaction that caused the error.

Default

Do not autostart

Syntax

```
AUTOSTART {{EXTRACT | REPLICAT | ER} group_name | JAGENT}
```

EXTRACT
Starts Extract automatically.

REPLICAT
Starts Replicat automatically.

ER
Starts Extract and Replicat automatically.

JAGENT
Starts the Oracle GoldenGate Monitor JAgent automatically. For more information, see *Administering Oracle GoldenGate Monitor*.

group_name
Valid for EXTRACT, REPLICAT, ER only. JAGENT does not take a group name as input. Specifies a group name or wildcard specification for multiple groups. When wildcarding is used, Oracle GoldenGate starts all groups of the specified process type that satisfy the wildcard on the local system, except those in PASSIVE mode.

Example

```
AUTOSTART ER *
```

7.5 BOOTDELAYMINUTES

Valid For

Manager

Description

Use the BOOTDELAYMINUTES parameter on a Windows system to delay the activities that Manager performs when it starts, such as executing parameters. For example, BOOTDELAYMINUTES can be used to delay AUTOSTART parameters until database services are started.

Specify BOOTDELAYMINUTES before other parameter entries. This parameter only supports Windows.

Default

None (no delay)

Syntax

```
BOOTDELAYMINUTES minutes
```

minutes
The number of minutes to delay after system startup before starting Oracle GoldenGate processing.

Example

```
BOOTDELAYMINUTES 5
```

7.6 CHARSET

Valid For

Extract, Replicat, DEFGEN, Manager, and GLOBALS

Description

Use the CHARSET parameter to specify the character set of the parameter files in the local Oracle GoldenGate instance. By default, the parameter file is created in the default character set of the local operating system. CHARSET specifies an alternative character set to use in the event that the local platform does not support a required character or characters.

CHARSET cannot be used with query parameters.

CHARSET allows operating-system incompatible characters, including multi byte characters, to be used in the parameter file without the need for an escape sequence (\uXXXX) when the local platform does not support multibyte characters as the default character set of the operating system.

CHARSET can also be used when the parameter file is being created on one system but will be used on a different system with a different character set. To avoid possible incompatibilities between different character sets, you should create parameter files on the same system where they will be used by Oracle GoldenGate.

> **Note:**
>
> Use of CHARSET in the mgr.prm file is not supported in 12.1.2.x or earlier releases.

Placement in the Parameter File

CHARSET must be placed on the first line of the parameter file.

Usage in the GLOBALS File

CHARSET in a GLOBALS file sets a default character set for the parameter files of all local processes. CHARSET in an individual parameter file overrides the default that is set in GLOBALS.

Usage in Nested Parameter Files

You can use CHARSET in a parameter file that includes an OBEY or INCLUDE parameter, but the referenced parameter file does not inherit the CHARSET character set. The CHARSET character set is used to read wildcarded object names in the referenced file, but you must use an escape sequence (\uXXXX) to specify all other incompatible characters in the referenced file.

Default

None

Syntax

```
CHARSET character_set
```

character_set
Any supported character set.

Example

```
CHARSET UTF-8
```

7.7 CHECKMINUTES

Valid For

Manager

Description

Use the CHECKMINUTES parameter to control how often Manager performs maintenance activities. Decreasing this parameter can significantly affect performance if trail files roll over frequently. Other events, such as processes ending abnormally, also trigger the maintenance cycle.

Default

Every 10 minutes

Syntax

```
CHECKMINUTES minutes
```

minutes
The frequency, in minutes, to perform maintenance activities.

Example

```
CHECKMINUTES 15
```

7.8 COMMENT | --

Valid For

Manager, Extract, Replicat

Description

Use the `COMMENT` parameter or double hyphens (--) to indicate comments anywhere within a parameter file. Anything on the same line after `COMMENT` or double hyphens is ignored during processing. Comments that continue to the next line must be preceded by another `COMMENT` keyword or double hyphens.

> **Note:**
>
> If any columns in the tables that are being synchronized contain the word "`comment`," there may be conflicts with the `COMMENT` parameter. Use double hyphens instead.

`COMMENT` cannot be used with query parameters.

See *Administering Oracle GoldenGate* for more information about working with parameter files.

Default

None

Syntax

```
{COMMENT comment} | {-- comment}
```

Examples

Example 1

```
COMMENT GoldenGate param file for fin Extract group.
```

Example 2

```
-- GoldenGate param file for fin Extract group.
```

7.9 DOWNREPORT

Valid For

Manager

Description

Use the DOWNREPORTMINUTES or DOWNREPORTHOURS parameter to specify the frequency with which Manager reports Extract and Replicat processes that are not running. Whenever a process starts or stops, events are generated to the error log, and those messages can easily be overlooked if the log is large. DOWNREPORTMINUTES and DOWNREPORTHOURS report on a periodic basis to ensure that you are aware of stopped processes.

If DOWNREPORT is explicitly indicated and the value of the CHECKMINUTES parameter is greater than that of DOWNREPORT, then CHECKMINUTES acquires the value of DOWNREPORT.

To report on running processes, use the UPREPORT parameter.

Default

Do not report down processes.

Syntax

```
DOWNREPORTMINUTES minutes | DOWNREPORTHOURS hours
```

minutes
The frequency, in minutes, to report processes that are not running. The minimum is 0.

hours
The frequency, in hours, to report processes that are not running. The minimum is 0.

Example

The following generates a report every 30 minutes.

```
DOWNREPORTMINUTES 30
```

7.10 DYNAMICPORTLIST

Valid For

Manager

Description

Use the DYNAMICPORTLIST parameter to specify a list of available ports to which the following local Oracle GoldenGate processes can bind for communication with a remote Oracle GoldenGate process:

- Collector: to communicate with a remote Extract to receive incoming data.
- Replicat: to communicate with a remote Extract to receive data during an initial load task.

- Passive Extract: to communicate with a remote Collector
- GGSCI: to issue remote commands

Specify enough ports to accommodate expansion of the number of processes without having to stop and restart Manager to add them to the list. You can specify an individual port, a range of ports, or both.

Default

None

Syntax

```
DYNAMICPORTLIST {port | port-port} [ , ...]
```

port
A port number that can be allocated. The maximum number of port entries is 5000.

- To specify multiple ports, use a comma-delimited list. Example:

  ```
  7830, 7833
  ```

- To specify a range of ports, use a dash (-) to separate the first and last port in the range. Do not put any spaces before or after the dash. Example:

  ```
  7830-7835
  ```

- To specify a range of ports plus an individual port, place a comma between the range and the individual port number. Example:

  ```
  7830-7835, 7839
  ```

Example

```
DYNAMICPORTLIST 7820-7830, 7833, 7835
```

7.11 MONITORING_HEARTBEAT_TIMEOUT

Valid For

Manager

Description

Use `MONITORING_HEARTBEAT_TIMEOUT` to set a process as non-responsive in a specified number of seconds.

Default

10 seconds.

Syntax

```
MONITORING_HEARTBEAT_TIMEOUT seconds
```

seconds
Specifies the time interval, in seconds, for Manager to set processes as non-responsive. The minimum is 10 seconds and the maximum is 60.

Examples

```
MONITORING_HEARTBEAT_TIMEOUT 20
```

7.12 LAGCRITICAL

Valid For

Manager

Description

Use the LAGCRITICALSECONDS, LAGCRITICALMINUTES, or LAGCRITICALHOURS parameter to specify a lag threshold that is considered critical, and to force a warning message to the error log when the threshold is reached. This parameter affects Extract and Replicat processes on the local system.

Default

Do not report lag information

Syntax

LAGCRITICALSECONDS *seconds* | LAGCRITICALMINUTES *minutes* | LAGCRITICALHOURS *hours*

LAGCRITICALSECONDS *seconds*
Sets the critical lag threshold in seconds. The minimum is 0.

LAGCRITICALMINUTES *minutes*
Sets the critical lag threshold in minutes. The minimum is 0.

LAGCRITICALHOURS *hours*
Sets the critical lag threshold in hours. The minimum is 0.

Example

```
LAGCRITICALSECONDS 60
```

7.13 LAGINFO

Valid For

Manager

Description

Use the LAGINFOSECONDS, LAGINFOMINUTES, or LAGINFOHOURS parameter to specify a basic lag threshold; if lag exceeds the specified value, Oracle GoldenGate reports lag information to the error log. If the lag exceeds the value specified with the LAGCRITICAL parameter, Manager reports the lag as critical; otherwise, it reports the lag as an informational message. A value of zero (0) forces a message at the frequency specified with the LAGREPORTMINUTES or LAGREPORTHOURS parameter.

Default

Do not report lag information

Syntax

```
LAGINFOSECONDS seconds | LAGINFOMINUTES minutes | LAGINFOHOURS hours
```

LAGINFOSECONDS *seconds*
Sets a basic lag threshold in seconds. The minimum is 0.

LAGINFOMINUTES *minutes*
Sets a basic lag threshold in minutes. The minimum is 0.

LAGINFOHOURS *hours*
Sets a basic lag threshold in hours. The minimum is 0.

Example

In this example, Oracle GoldenGate reports lag when it exceeds one hour.

```
LAGINFOHOURS 1
```

7.14 LAGREPORT

Valid For

Manager

Description

Use the LAGREPORTMINUTES or LAGREPORTHOURS parameter to specify the interval at which Manager checks for Extract and Replicat lag. Use of this parameter also requires the use of the LAGINFO and LAGCRITICAL parameters. If LAGREPORT is not specified, lag is not reported.

If LAGREPORT is used and the value of the CHECKMINUTES parameter is greater than LAGREPORT, then CHECKMINUTES will acquire the value of LAGREPORT.

Default

None

Syntax

```
LAGREPORTMINUTES minutes | LAGREPORTHOURS hours
```

LAGREPORTMINUTES *minutes*
The frequency, in minutes, to check for lag. The minimum is 0.

LAGREPORTHOURS *hours*
The frequency, in hours, to check for lag. The minimum is 0.

Example

```
LAGREPORTHOURS 1
```

7.15 PURGEMARKERHISTORY

Valid For

Manager

Description

Use the PURGEMARKERHISTORY parameter to control the size of the Oracle GoldenGate marker table. This parameter purges rows that are not needed any more. You can purge the marker table at any time. This parameter is only valid for an Oracle database in which the capture configuration uses the Oracle GoldenGate DDL trigger to support DDL replication. For more information about the Oracle GoldenGate DDL objects, see Configuring DDL Support.

To determine the name of the marker table, Oracle GoldenGate first looks for a name specified with the MARKERTABLE parameter in the GLOBALS file. If that parameter does not exist, Oracle GoldenGate uses the default name of GGS_MARKER.

You can specify maximum and minimum lengths of time to keep a row, based on the last modification date. Both maximum and minimum rules must be specified; otherwise Manager does not have complete criteria for when to delete the row. For example, MINKEEPHOURS 3 used with MAXKEEPHOURS 5 specifies to keep rows that have not been modified in the past three hours, but delete them when they have not been modified for at least five hours.

PURGEMARKERHISTORY requires a logon to be specified with the USERID or USERIDALIAS parameter and, depending on the type of database, the SOURCEDB parameter.

Default

Purge every hour

Syntax

```
PURGEMARKERHISTORY
{, max_rule}
[, min_rule]
[, frequency]
```

max_rule
Required. Can be one of the following to set the maximum amount of time to keep rows.

MAXKEEPHOURS *n*
Purges if the row has not been modified for *n* number of hours. The minimum is 1 and the maximum is 1000.

MAXKEEPDAYS n
Purges if the row has not been modified for *n* number of days. The minimum is 1 and the maximum is 365.

min_rule
Can be one of the following to set the minimum amount of time to keep rows.

MINKEEPHOURS *n*
Keeps an unmodified row for at least the specified number of hours. The minimum is 1 and the maximum is 1000.

MINKEEPDAYS *n*
Keeps an unmodified row for at least the specified number of days. The minimum is 1 and the maximum is 365.

frequency
Sets the frequency with which to purge marker history. The default interval at which Manager evaluates potential maintenance tasks is 10 minutes, as specified with the `CHECKMINUTES` parameter. At that interval, Manager evaluates the `PURGEMARKERHISTORY` frequency and conducts the purge at the specified *`frequency`*. *`frequency`* can be one of the following:

FREQUENCYMINUTES *n*
Sets the frequency, in minutes, with which to purge marker history. The default purge frequency is 60 minutes. The minimum is 1 and the maximum is 360.

FREQUENCYHOURS *n*
Sets the frequency, in hours, at which to purge marker history. See "CHECKMINUTES" for more information about controlling the interval between Manager maintenance checks. The minimum is 1 and the maximum is 24.

Example

The following example keeps all rows that have not been modified in the past three days and deletes them when they have not been modified for at least five days. The purge frequency is 30 minutes.

```
PURGEMARKERHISTORY MINKEEPDAYS 3, MAXKEEPDAYS 5, FREQUENCYMINUTES 30
```

7.16 PORT

Valid For

Manager

Description

Use the `PORT` parameter to specify a TCP/IP port number for the Manager process on which to interact with remote processes that request dynamic services, typically either an initial-load Replicat or the Collector process. Use the default port number when possible. The minimum is 1 and the maximum is 65535.

Default

Port 7809

Syntax

```
PORT number
```

number
An available port number.

Example

```
PORT 7809
```

7.17 SOURCEDB

Valid For

Manager, Extract, DEFGEN

Description

Use the SOURCEDB parameter for databases or data sets that require a data source name or identifier to be specified explicitly as part of the connection information. This option is required to identify one of the following:

- The source database for heterogeneous databases.
- The data source name (DSN) for supported databases that use ODBC
- The default DB2 for i database.

Tables specified in TABLE statements that follow SOURCEDB are assumed to be from the specified data source.

You might need to use the USERID or USERIDALIAS parameter in the SOURCEDB parameter statement, depending on the authentication that is required for the data source.

For databases that allow authentication at the operating-system level, you can specify SOURCEDB without USERID or USERIDALIAS.

For DB2 LUW, the SOURCEDB statement must refer to the database by its real name, rather than by any alias.

See USERID | NOUSERID or USERIDALIAS for more information.

Default

None

Syntax

```
SOURCEDB data_source[, SESSIONCHARSET character_set]
```

data_source
The name of the database, catalog, or data source name as applicable for the database.
For MySQL databases, you can use the format of SOURCEDB *database_name@host_name* to avoid connection issues caused by the incorrect configuration of localhost in the local hosts file. If running MySQL on a port other than the default of 3306, you must specify the port number in the connect string: SOURCEDB *database_name@host_name*:*port*.

SESSIONCHARSET *character_set*
Supports MySQL. Sets the database session character set for the process login session. This parameter overrides any SESSIONCHARSET that is specified in the GLOBALS file.

Examples

Example 1
This example shows SOURCEDB using OS-level authentication.

```
SOURCEDB mydb
```

Example 2
This example shows SOURCEDB with the USERIDALIAS parameter.

```
SOURCEDB mydb, USERIDALIAS tiger1
```

7.18 PURGEDDLHISTORY | PURGEDDLHISTORYALT

Valid For

Manager

Description

Use the PURGEDDLHISTORY and PURGEDDLHISTORYALT parameters to control the size of the DDL history tables that support DDL capture. These tables are created in an Oracle database to support trigger-based DDL capture.

These parameters cause Manager to purge rows that are not needed any more. You can specify the maximum and minimum amount of time to keep a row, based on the last modification date. Both maximum and minimum rules must be specified; otherwise Manager does not have a complete criteria for when to delete the row. For example, MINKEEPHOURS 3 used with MAXKEEPHOURS 5 specifies to keep rows that have not been modified in the past three hours, but to delete them when they have not been modified for at least five hours.

These parameters require a logon to be specified with the USERID or USERIDALIAS parameter.

> **WARNING:**
>
> Use caution when purging the history tables. They are critical to the integrity of the DDL synchronization processes. Premature purges are non-recoverable through Oracle GoldenGate. To prevent any possibility of permanent DDL data loss, make regular backups of the Oracle GoldenGate DDL schema.

Default

Purge every hour

Syntax

```
PURGEDDLHISTORY | PURGEDDLHISTORYALT
{, max_rule}
[, min_rule]
[, frequency]
```

PURGEDDLHISTORY
Purges the DDL history table. This table tracks DDL operations. To determine the name of the history table to purge, Oracle GoldenGate first looks for a name specified with the DDLTABLE parameter in the GLOBALS file. If that parameter does not exist, Oracle GoldenGate uses the default name of GGS_DDL_HIST.

PURGEDDLHISTORYALT
Purges the internal DDL history table. This table tracks partitioned object IDs that are associated with the object ID of a table. To determine the name of the internal history table to purge, Oracle GoldenGate first looks for a name specified with the DDLTABLE parameter in the GLOBALS file and appends _ALT to it. If that parameter does not exist, Oracle GoldenGate uses the default name of GGS_DDL_HIST_ALT.

max_rule
Required. Can be one of the following to set the maximum amount of time to keep rows.

MAXKEEPHOURS *n*
Purges if the row has not been modified for *n* number of hours. The minimum is 1 and the maximum is 1000.

MAXKEEPDAYS n
Purges if the row has not been modified for *n* number of days. The minimum is 1 and the maximum is 365.

min_rule
Can be one of the following to set the minimum amount of time to keep rows.

MINKEEPHOURS *n*
Keeps an unmodified row for at least the specified number of hours. The minimum is 1 and the maximum is 1000.

MINKEEPDAYS *n*
Keeps an unmodified row for at least the specified number of days. The minimum is 1 and the maximum is 365.

frequency
Sets the frequency with which to purge DDL history. The default interval at which Manager evaluates potential maintenance tasks is 10 minutes, as specified with the CHECKMINUTES parameter. At that interval, Manager evaluates the PURGEDDLHISTORY or PURGEDDLHISTORYALT frequency and conducts the purge at the specified *frequency*. *frequency* can be one of the following:

FREQUENCYMINUTES *n*
Sets the frequency, in minutes, with which to purge DDL history. The default purge frequency is 60 minutes. The minimum is 1 and the maximum is 360.

FREQUENCYHOURS *n*
Sets the frequency, in hours, at which to purge DDL history.
See "CHECKMINUTES" for more information about controlling the interval between Manager maintenance checks. The minimum is 1 and the maximum is 24.

Example

The following example keeps all rows that have not been modified in the past three days and deletes them when they have not been modified for at least five days. The purge frequency is 30 minutes.

```
PURGEDDLHISTORY MINKEEPDAYS 3, MAXKEEPDAYS 5, FREQUENCYMINUTES 30
```

7.19 PURGEOLDEXTRACTS for Extract and Replicat

Valid For

Extract and Replicat

Description

Use the PURGEOLDEXTRACTS parameter in an Extract or Replicat parameter file to delete old trail files whenever Oracle GoldenGate starts processing from a new one. Preventing the accumulation of trail files conserves disk space. Purges are conducted after the process is done with the file as indicated by checkpoints.

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 Oracle 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
```

7.20 PURGEOLDEXTRACTS for Manager

Valid For

Manager

Description

Use the PURGEOLDEXTRACTS parameter in a Manager parameter file to purge trail files when Oracle GoldenGate has finished processing them. Without using

PURGEOLDEXTRACTS, no purging is performed, and trail files can consume significant disk space.

Using PURGEOLDEXTRACTS as a Manager parameter is recommended rather than using the Extract or Replicat version of PURGEOLDEXTRACTS. As a Manager parameter, PURGEOLDEXTRACTS allows you to manage trail files in a centralized fashion and take into account multiple processes.

How to Use PURGEOLDEXTRACTS for Manager

To control the purging, follow these rules:

- USECHECKPOINTS triggers a purge when all processes are finished with a file as indicated by their checkpoints. Basing the purges on checkpoints ensures that Manager does not delete any data until all processes are finished with it. This is essential in a production environment to ensure data integrity. USECHECKPOINTS considers the checkpoints of both Extract and Replicat before purging. Because USECHECKPOINTS is the default, it need not be specified in the PURGEOLDEXTRACTS statement. Manager obeys USECHECKPOINTS unless there is an explicit NOUSECHECKPOINTS entry.
- Use the MINKEEP rules to set a minimum amount of time to keep data:
  - Use MINKEEPHOURS or MINKEEPDAYS to keep data for *n* hours or days.
  - Use MINKEEPFILES to keep at least *n* trail files including the active file. The default number of files to keep is 1.

  Use only one of the MINKEEP options. If more than one is used, Oracle GoldenGate selects one of them based on the following:
  - If both MINKEEPHOURS and MINKEEPDAYS are specified, only the last one is accepted, and the other is ignored.
  - If either MINKEEPHOURS or MINKEEPDAYS is used with MINKEEPFILES, then MINKEEPHOURS or MINKEEPDAYS is accepted, and MINKEEPFILES is ignored.

Manager evaluates potential maintenance tasks based on the value set for the CHECKMINUTES parameter. When that value is reached, Manager determines which files to purge based on the 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. The following are possible PURGEOLDEXTRACT rule combinations and the actions that Manager takes for them:

- USECHECKPOINTS without MINKEEP rules: If checkpoints indicate that a file has been processed completely, it will be purged unless doing so would violate the default rule to keep at least one file.
- USECHECKPOINTS with MINKEEP rules: If checkpoints indicate that a file has been processed completely, it will be purged unless doing so would violate the MINKEEP rules.
- NOUSECHECKPOINTS without MINKEEP rules: The checkpoints are not considered, and the file will be purged unless doing so would violate the default rule to keep at least one file.
- NOUSECHECKPOINTS with MINKEEP rules: A file will be purged unless doing so would violate the MINKEEP rules.

Additional Guidelines for PURGEOLDEXTRACTS for Manager

- Do not use more than 500 PURGEOLDEXTRACTS parameter statements in the same Manager parameter file.
- When using this parameter, do not permit trail files to be deleted by any user or program other than Oracle GoldenGate. It will cause PURGEOLDEXTRACTS to function improperly.
- When trails are stored on NFS, there is a difference in system time between the NFS drive and the local system where Manager is running. The trail is created with the NFS time, but the timestamps of the records in the trail are compared with the local system time to determine whether to purge them or not. Take into account any time differences when you create your MINKEEP rules.

Default

```
USECHECKPOINTS
```

Syntax

```
PURGEOLDEXTRACTS trail
[, USECHECKPOINTS | NOUSECHECKPOINTS]
[, MINKEEP_rule]
[, frequency]
```

trail
The trail to purge. Use a relative or fully qualified name.

USECHECKPOINTS
Allows purging according to any MINKEEP rules after all Extract and Replicat processes are done with the data as indicated by checkpoints. When using USECHECKPOINTS, you can use the USERID or USERIDALIAS parameters in the Manager parameter file, so that Manager can query the Replicat checkpoint table to get checkpoint information though it is not required.

NOUSECHECKPOINTS
Allows purging without considering checkpoints, based either on the default rule to keep a minimum of one file (if no MINKEEP rule is used) or the number of files specified with a MINKEEP rule.

MINKEEP_rule
Can be one of the following to set rules for the minimum amount of time to keep an inactive file.

MINKEEPHOURS *n*
Keeps an inactive file for at least the specified number of hours. The minimum is 1 and the maximum is 1000.

MINKEEPDAYS *n*
Keeps an inactive file for at least the specified number of days. The minimum is 1 and the maximum is 365.

MINKEEPFILES *n*
Keeps at least *n* trail files, including the active file. The minimum is 1 and the maximum is 100. The default is 1.

frequency
Sets the frequency with which to purge inactive trail files. The default time for Manager to evaluate potential maintenance tasks is 10 minutes, as specified with the CHECKMINUTES parameter. At that interval, Manager evaluates the PURGEOLDEXTRACTS frequency and conducts the purge after the specified *frequency*.
frequency can be one of the following:

FREQUENCYMINUTES *n*
Sets the frequency in minutes. The default purge frequency is 60 minutes. The minimum is 1 and the maximum is 360.

FREQUENCYHOURS *n*
Sets the frequency in hours. The minimum is 1 and the maximum is 24.

See "CHECKMINUTES" for more information about controlling the Manager maintenance check interval.

Examples

Example 1
Status: Trail files AA000000, AA000001, and AA000002 exist. Replicat has been stopped for four hours and is not finished processing any of the files. The Manager parameters include:

```
PURGEOLDEXTRACTS /ggs/dirdat/AA*, USECHECKPOINTS, MINKEEPHOURS 2
```

Result: The amount of time that files must be retained was exceeded, but no files will be purged because checkpoints indicate that Replicat is not finished processing them.

Example 2
Status: Trail files AA000000, AA000001, and AA000002 exist. Replicat has been stopped for four hours and is not finished processing any of the files. The Manager parameters include:

```
PURGEOLDEXTRACTS /ggs/dirdat/AA*, NOUSECHECKPOINTS, MINKEEPHOURS 2
```

Result: All of the trail files will be purged because the minimum time to keep them was satisfied, and checkpoints are not considered before purging.

Example 3
Status: Replicat and Extract are finished processing data. There has been no access to the trail files for the last five hours. Trail files AA000000, AA000001, and AA000002 exist. The Manager parameters include:

```
PURGEOLDEXTRACTS /ggs/dirdat/AA*, USECHECKPOINTS, MINKEEPHOURS 4, &
MINKEEPFILES 4
```

Result: This is an example of why only one of the MINKEEP options should be set. USECHECKPOINTS requirements were satisfied, so the MINKEEP rules are considered when determining whether to purge AA000002. Only two files will remain if AA000002 is purged, and that violates the MINKEEPFILES rule. Because both MINKEEPFILES and MINKEEPHOURS are specified, however, MINKEEPFILES is ignored. The file will be purged because it has not been accessed for five hours, and that satisfies the MINKEEPHOURS requirement of four hours.

7.21 PURGEOLDTASKS

Valid For

Manager

Description

Use the PURGEOLDTASKS parameter to purge Extract and Replicat tasks after a specific amount of time or after 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, 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.

No more than 300 PURGEOLDTASKS parameter statements may be used in the same Manager parameter file.

Default

None

Syntax

```
PURGEOLDTASKS {EXTRACT | REPLICAT | ER} group
{AFTER number {DAYS | HOURS} | USESTOPSTATUS}
```

EXTRACT | REPLICAT | ER
The process for which you want to purge tasks. Use the ER option to specify both Extract and Replicat process types.

group
The group name or a wildcard to specify multiple groups.

AFTER *number* {DAYS | HOURS}
Purges if the task has not been updated for a specified number of days or hours.

USESTOPSTATUS
Purges if the task was stopped gracefully or never was started.

Example

The following example deletes all Extract tasks that have not been updated for at least three days, and it deletes the test_rep Replicat task if it stopped gracefully and has not been updated for at least two hours.

```
PURGEOLDTASKS EXTRACT *, AFTER 3 DAYS
PURGEOLDTASKS REP test_rep, AFTER 2 HOURS, USESTOPSTATUS
```

7.22 STARTUPVALIDATIONDELAY[CSECS]

Valid For

Manager

Description

Use the STARTUPVALIDATIONDELAY or STARTUPVALIDATIONDELAYCSECS parameter to set a delay time after which Manager validates the status of a process that was started with the START EXTRACT or START REPLICAT command. If a process is not running after the specified delay time, an error message is displayed at the GGSCI prompt.

These parameters account for processes that fail before they can generate an error message or report, for example when there is not enough memory to launch the processes. Startup validation makes Oracle GoldenGate users aware of such failures. The minimum is 0.

Default

0 seconds (do not validate startup status)

Syntax

```
STARTUPVALIDATIONDELAY seconds | STARTUPVALIDATIONDELAYCSECS centiseconds
```

STARTUPVALIDATIONDELAY *seconds*
Specifies the delay in seconds.

STARTUPVALIDATIONDELAYCSECS *centiseconds*
Specifies the delay in centiseconds.

Example

In the following example, Manager waits ten centiseconds after a START command is issued and then checks the status of the process.

```
STARTUPVALIDATIONDELAYCSECS 10
```

7.23 USERIDALIAS

Valid For

Manager, Extract, Replicat, DEFGEN

Supported for

DB2 for i

DB2 LUW

DB2 on z/OS

MySQL

Oracle

SQL Server

Teradata

Description

Use the USERIDALIAS parameter to specify authentication for an Oracle GoldenGate process to use when logging into a database. The use of USERIDALIAS requires the use of an Oracle GoldenGate credential store. Specify USERIDALIAS before any TABLE or MAP entries in the parameter file.

> **Note:**
>
> The privileges that are required for the USERIDALIAS user vary by database. See the *Using Oracle GoldenGate for Heterogeneous Databases*to determine the privileges that are required for the Oracle GoldenGate database users.

USERIDALIAS Compared to USERID

USERIDALIAS enables you to specify an alias, rather than a user ID and password, in the parameter file. The user IDs and encrypted passwords are stored in a credential store. USERIDALIAS supports databases running on Linux, UNIX, and Windows platforms.

USERID requires either specifying the clear-text password in the parameter file or encrypting it with the ENCRYPT PASSWORD command and, optionally, storing an encryption key in an ENCKEYS file. USERID supports a broad range of the databases that Oracle GoldenGate supports. In addition, it supports the use of an operating system login for Oracle databases.

USERIDALIAS Requirements Per Database Type

The usage of USERIDALIAS varies depending on the database type.

> **Note:**
>
> Logins that require a database user and password must be stored in the Oracle GoldenGate credential store.

DB2 for LUW

Use USERIDALIAS with the SOURCEDB or TARGETDB parameter for all Oracle GoldenGate processes that connect to a DB2 LUW database using database authentication. You can omit USERIDALIAS and only use SOURCEDB or TARGETDB if the database is configured allow authentication at the operating-system level. See "SOURCEDB" and "TARGETDB" for more information.

MySQL

Use USERIDALIAS for all Oracle GoldenGate processes that connect to a MySQL database. The SOURCEDB or TARGETDB parameter is not required.

Oracle

Use USERIDALIAS for Oracle GoldenGate processes that connect to an Oracle database.

- The SOURCEDB or TARGETDB parameter is not required.
- Specify the alias of a database credential that is stored in the Oracle GoldenGate credential store.
- (*Oracle Enterprise Edition earlier than 11.2.0.2*) Special database privileges are required for the USERIDALIAS user when Extract is configured to use LOGRETENTION. These privileges might have been granted when Oracle GoldenGate was installed. See the Log Retention Options in *Using Oracle GoldenGate for Oracle Database* for more information about LOGRETENTION.
- (*Oracle Standard or Enterprise Edition 11.2.0.2 or later*) To use USERIDALIAS for an Extract group that is configured for integrated capture, the user must have the privileges granted in the dbms_goldengate_auth.grant_admin_privilege procedure, and the user must be the same one that issues DBLOGIN and REGISTER EXTRACT or UNREGISTER EXTRACT for the Extract group that is associated with this USERIDALIAS.
- To support capture from an Oracle container database, the user that is specified with USERID must log on to the root container and must be a common database user. A connect string must be supplied for this user, for example: C##GGADM@FINANCE. For more information, see Configuring the Primary Extract in *Using Oracle GoldenGate for Heterogeneous Databases*.

SQL Server

Use USERIDALIAS if the ODBC data source connection that will be used by the Oracle GoldenGate process is configured to supply database authentication. USERIDALIAS can be a specific login that is assigned to the process or any member of an account in the System Administrators or Server Administrators fixed server role.

- On a source SQL Server system, also use the SOURCEDB parameter to specify the source ODBC data source.
- On a target SQL Server system, also use the TARGETDB parameter to specify the target ODBC data source.

Teradata

Use USERIDALIAS for Oracle GoldenGate processes that connect to a Teradata database.

- On a source Teradata system, also use the SOURCEDB parameter to specify the source ODBC data source.
- On a target Teradata system, also use the TARGETDB parameter to specify the target ODBC data source.

Default

None

Syntax

```
USERIDALIAS alias [DOMAIN domain] [SYSDBA]
[, THREADS (threadID[, threadID][, ...][, thread_range[, thread_range][, ...])]
```

alias
Specifies the alias of a database user credential that is stored in the Oracle GoldenGate credential store. Refer to USERID Requirements Per Database Type for additional guidelines.

DOMAIN *domain*
Specifies the credential store domain for the specified alias. A valid domain entry must exist in the credential store for the specified alias.

SYSDBA
(Oracle) Specifies that the user logs in as `sysdba`.

THREADS (*threadID*[, *threadID*][, ...][, *thread_range*[, *thread_range*][, ...])
Valid for Replicat. Links the specified credential to one or more threads of a coordinated Replicat. Enables you to specify different logins for different threads.

***threadID*[, *threadID*][, ...]**
Specifies a thread ID or a comma-delimited list of threads in the format of `threadID, threadID, threadID`.

[, *thread_range*[, *thread_range*][, ...]
Specifies a range of threads in the form of `threadIDlow-threadIDhigh` or a comma-delimited list of ranges in the format of `threadIDlow-threadIDhigh, threadIDlow-threadIDhigh`.

A combination of these formats is permitted, such as `threadID, threadID, threadIDlow-threadIDhigh`.

Examples

Example 1
The following supplies a credential for the user in the credential store that has the alias of `tiger1` in the domain of `east`.

Example 2
The following supplies a credential for thread 3 of a coordinated Replicat.

```
USERIDALIAS tiger1 DOMAIN east THREADS (3)
```

7.24 VERIDATAREPORTAGE

Valid For

Manager

Description

Use the `VERIDATAREPORTAGE` parameter to purge old Veridata (`veriagt`)report files when they have reached the age limit.

Default

7 DAYS

Syntax

VERIDATAREPORTAGE *number* [*time units*]

number
A positive integer.

time units

DAYS|DAY|HOURS|HOUR|HRS|MINUTES|MINUTE|MIN|SECONDS|SECOND|SEC

Example

```
VERIDATAREPORTAGE 5 HOURS
```

8
Oracle GoldenGate Native Commands

This chapter contains reference information for commands that can be used to control Oracle GoldenGate from the native command line of the operating system (outside the GGSCI command interface). This enables you to include the commands in scripts to run programs non-interactively.

Topics:

- Summary of Oracle GoldenGate IBM i Native Commands
- checkprm
- convchk
- defgen
- extract
- install
- keygen
- logdump
- mgr
- replicat

8.1 Summary of Oracle GoldenGate IBM i Native Commands

This section summarizes the commands that can be issued directly from the native command line of the Linux, UNIX, Windows, or IBM i platforms. For the purpose of this document, commands issued from the IBM i PASE environment are considered UNIX commands.

On the IBM i platform, these commands are stored in the Oracle GoldenGate installation library and can be used instead of issuing them from a PASE environment. With this support, it is possible to use the typical job submission tools such as SBMJOB to operate the Oracle GoldenGate product. If submitted to batch, the output is written to a spool file, and only job messages and any exceptions are written to the job log. In a typical installation, a batch submitted command should have both a `QPRINT` output spool file and a joblog spool file.

To use the native commands from the IBM CLI, you need only include the Oracle GoldenGate installation library in the library list, or reference it explicitly through a qualified name such as *`OGGLIB`*/GGSCI. During the execution of the command the current directory will be set to the Oracle GoldenGate installation directory, and all appropriate environment variables will be set to operate the Oracle GoldenGate commands. Therefore, the paths for any parameter that can take a path may be specified either as an absolute path name or as a relative path name based at the Oracle GoldenGate installation directory. The `OTHERS` parameter of the IBM i CLI

commands is used to allow the specification of other parameters not explicitly exposed by the IBM i CLI commands. For example, if you want to specify `REPORTFILE` and `PROCESSID` for Extract, you would use the following syntax:

```
EXTRACT PARAMFILE('dirprm/myext.prm') OTHERS(REPORTFILE 'dirrpt/myext.rpt'
PROCESSID myext)
```

> **Note:**
>
> Normally, Extract and Replicat should be run from GGSCI, but some situations, such as certain initial load procedures, require running them from the command line of the operating system.

Table 8-1 Commands

Command	Description
checkprm	Assess the validity of the specified parameter file.
convchk	Converts the trail files from 9 digit to 6 digit checkpoint record for the named Extract group.
defgen	Runs the DEFGEN program. This program generates data definitions in a file that is transferred to a remote system to support the retrieval of object metadata.
extract	Runs the Extract program. This program captures either full data records or transactional data changes, depending on configuration parameters, and then sends the data to a trail for further processing by a downstream process, such as a data-pump Extract or the Replicat process.
GGSCI Command Interface	Runs the Oracle GoldenGate command line interface.
keygen	Runs the KEYGEN utility. This utility generates encryption keys to support Oracle GoldenGate security.

Table 8-1 (Cont.) Commands

Command	Description
logdump	Runs the Logdump utility. This utility enables the contents of a trail file to be viewed for the purpose of troubleshooting. This command takes no arguments. For more information about Logdump, see Logdump Reference for Oracle GoldenGate.
mgr	Runs the Manager program. Manager is the parent process of Oracle GoldenGate and is responsible for the management of its processes and files, resources, user interface, and the reporting of thresholds and errors.
replicat	Runs the Replicat program. This program reads a trail, performs mapping and manipulation as needed, and applies the data to a target database.

8.2 checkprm

Use the `checkprm` command to assess the validity of the specified parameter file, with a configurable application and running environment. It can provide either a simple `PASS/FAIL` or with optional details about how the values of each parameter are stored and interpreted.

When you use `checkprm` and do not use any of these arguments, then `checkprm` attempts to automatically detect Extract or Replicat and the platform and database of the Oracle GoldenGate installation.

For more information about using `checkprm`, see *Administering Oracle GoldenGate*.

Syntax

```
checkprm
[None]
[-v]
[? | help]
[parameter_file]
[-COMPONENT | -C) component_name]
[-MODE | -M) mode_name]
[-PLATFORM | -P) platform_name]
```

```
[-DATABASE | -D) database _ame]
[-VERBOSE | -V)]
```

None
Displays usage information.

-v
Displays banner. Cannot be combined with other options. Does not produce verbose (`-VERBOSE | -V`) output.

? | help
Displays detailed usage information, include all possible values of each option. Cannot be combine with other options.

parameter_file
Specifies the name of the parameter file, has to be the first argument if a validation is requested. You must specify the absolute path to the parameter file. For example, `CHECKPRM ./dirprm/myext.prm`.

-COMPONENT | -C *component_name*
Specifies the running component (application) that this parameter file is validated for. This option can be omitted for Extract or Replicat because automatic detection is attempted. Valid values include:

```
CACHEFILEDUMP COBGEN CONVCHK CONVPRM DDLCOB DEFGEN EMSCLNT EXTRACT GGCMD GGSCI
KEYGEN LOGDUMP
MGR OGGERR REPLICAT RETRACE
REVERSE SERVER GLOBALS
```

There is no default for this option.

-MODE | -M *mode_name*
Specifies the mode of the running application if applicable. This option is optional, only applicable to Extract or Replicat.
Valid input of this option includes:

- Classic Extract
- Integrated Extract
- Initial Load Extract
- Remote Task Extract
- Data Pump Extract
- Passive Extract
- Classic Replicat
- Coordinated Replicat
- Integrated Replicat
- Special Run Replicat
- Remote Task
- Replicat All

When key in the value for this option, the application name is optional, as long as it matches the value of component. For example, `"Â"Data Pump ExtractÂ"` is

equivalent to "Â"Data PumpÂ" if the component is Extract. However, it is invalid if the component is Replicat.

-PLATFORM | -P *platform_name*
Specifies the platform the application is supposed to run on. The default value is the platform that this checkprm executable is running on.
The possible values are:

```
AIX HP-OSS HPUX-IT HPUX-PA
Linux OS400 ZOS Solaris SPARC
Solaris x86 Windows x64 All
```

-DATABASE | -D *database_name*
Specifies the database the application is built against. The default value is the database for your Oracle GoldenGate installation.
The database options are:

```
Generic Oracle 8 Oracle 9i
Oracle 10g Oracle 11g Oracle 12c
DB2LUW 9.5 DB2LUW 9.7
DB2LUW 10.5 DB2LUW 10.1 DB2 Remote
Teradata MySQL  DB2 for I
DB2 for i Remote MS SQL DB2 z/OS
```

-VERBOSE | -V
Directs checkprm to display detailed parameter information, to demonstrate how the values are read and interpreted. It must be the last option specified in a validation and provides more information than the -v option.

8.3 convchk

Use the convchk command to convert trail files from 9 digit to 6 digit checkpoint record for the named extract group.

For more information about using convchk, see *Administering Oracle GoldenGate*.

Syntax for Windows, UNIX, and Linux

```
convchk <checkpoint_group> <trail_name> (SEQLEN_9D | SEQLEN_6D) [-force]
```

checkpoint_group
The name of the Extract group writing the trail.

trail_name
The relative or fully qualified path name of the trail that was used with the ADD EXTRAIL command or ADD RMTTRAIL command.

seqlen_9d
Sets the sequence length to 9 digits. This is the default.

seqlen_6d
Sets the sequence length to 6 digits.

-force
Optional, not recommended. It can be used if the Extract was not stopped gracefully.

8.4 defgen

Use `defgen` to run the DEFGEN utility from the command line of the Linux, UNIX, Windows, or IBM i operating system. The `defgen` command is installed in the Oracle GoldenGate installation directory or library.

For more information about using DEFGEN, see *Administering Oracle GoldenGate*.

Syntax for Windows, UNIX, and Linux

```
defgen paramfile parameter_file
[CHARSET character_set]
[COLCHARSET character_set]
[noextattr]
[pauseatend | nopauseatend]
[reportfile report_file]
```

The following syntax can also be used without any other options:

```
defgen defs_file updatecs charset
```

defgen
Used without options, the command runs the program interactively.

paramfile *parameter_file*
Required. Specifies the relative or absolute path name of the parameter file for the DEFGEN program that is being run.

CHARSET *character_set*
Any supported character set. See CHARSET for more information.

COLCHARSET *character_set*
Any supported character set. See COLCHARSET for more information.

noextattr
Can be used to support backward compatibility with Oracle GoldenGate versions that are older than Release 11.2.1 and do not support character sets other than ASCII, nor case-sensitivity or object names that are quoted with spaces. `NOEXTATTR` prevents DEFGEN from including the database locale and character set that support the globalization features that were introduced in Oracle GoldenGate Release 11.2.1. If the table or column name has multi-byte or special characters such as white spaces, DEFGEN does not include the table definition when `NOEXTATTR` is specified. If `APPEND` mode is used in the parameter file, `NOEXTATTR` is ignored, and the new table definition is appended in the existing file format, whether with the extra attributes or not.

pauseatend | nopauseatend
(Windows only) When the process stops, requires an Oracle GoldenGate user to look at the console output and then strike any key to clear it. Also indicates whether the process ended normally or abnormally.

reportfile *report_file*
Sends command output to the specified report file. Without the `reportfile` option, the command output is printed to the screen.

defs_file* updatecs *charset
Converts the character set of a definitions file to a different character set if the file is transferred to an operating system with an incompatible character set. This procedure takes the name of the definitions file and the targeted character set as input. For example: `defgen ./dirdef/source.def UPDATECS UTF-8`.
`updatecs` helps in situations such as when a Japanese table name on Japanese Windows is written in Windows CP932 to the data-definitions file, and then the definitions file is transferred to Japanese UNIX. The file cannot be used unless the UNIX is configured in PCK locale. Thus, you must use `updatecs` to convert the encoding of the definitions file to the correct format.

Syntax for IBM i CLI

```
DEFGEN PARAMFILE(input_file)
[OTHERS(other_options)]
```

PARAMFILE(*input_file*)
The input text file, known as an `OBEY` file, containing the commands that you want to issue, in the order they are to be issued, one command per line. The name can be anything supported by the operating system.

OTHERS(*other_options*)
Any options that are supported in the UNIX version of the command provided as a space separated list.

8.5 extract

Use `extract` to run the Extract program from the command line of the Linux, UNIX, Windows, or IBM i operating system. The `extract` command is installed in the Oracle GoldenGate installation directory or library.

Syntax for Windows, UNIX, and Linux

```
extract paramfile parameter_file
[atcsn CSN | aftercsn CSN]
[initialdataload]
[pauseatend | nopauseatend]
[processid PID]
[reportfile report_file]
```

extract
Used without options, the command runs the program interactively.

paramfile *parameter_file*
Required. Specifies the relative or absolute path name of the parameter file for the Extract program that is being run. The default location is the `dirprm` subdirectory of the Oracle GoldenGate installation directory.

atcsn *CSN* | aftercsn *CSN*
Starts the process at or after the specified commit sequence number (CSN).

initialdataload
Runs Extract to extract all of the data records directly from the source database to support an initial load to the target.

pauseatend | nopauseatend
(Windows only) When the process stops, requires an Oracle GoldenGate user to look at the console output and then strike any key to clear it. Also indicates whether the process ended normally or abnormally.

processid *PID*
A name for the process. This name must match the name that is specified for the `EXTRACT` parameter in the parameter file. Use one alphanumeric word. When used on IBM i, this name (up to the first 10 characters) will be used as the job name in the IBM i job list.

reportfile *report_file*
Sends command output to the specified report file. Without the `reportfile` option, the command output is printed to the screen. The default is the `dirrpt` subdirectory of the Oracle GoldenGate installation directory.

Syntax for IBM i CLI

```
EXTRACT PARAMFILE(input_file)
[OTHERS(other_options)]
```

PARAMFILE(*input_file*)
The input text file, known as an `OBEY` file, containing the commands that you want to issue, in the order they are to be issued, one command per line. The name can be anything supported by the operating system.

OTHERS(*other_options*)
Any options that are supported in the UNIX version of the command provided as a space separated list.

8.6 install

Use the `install` program from the command line of the Linux, UNIX, Windows, or IBM i operating system to:

- install event messages (into the registry) so they are displayed in the Windows Event Manager.
- install the Oracle GoldenGate Manager program as an NT service.

The `install` program is installed in the Oracle GoldenGate installation directory or library.

By default, the Manager service is installed to start automatically at system boot time. This can be changed by specifying the `MANUALSTART` option.

In addition, install can be used to de-install event messages and the Manager service.

Installation of event messages and the Manager program as a service is recommended. For example:

```
install item [item...]
```

In this command *item* is one of `addevents | deleteevents addservice | deleteservice autostart | manualstart` and the user credentials:

ADDEVENTS
Adds Oracle GoldenGate events. By default, the errors logged are generic. To display specific content, including the user name and process, the parameter file name, and the error text, copy the `category.dll` and `ggsmsg.dll` files from the installation directory to the `SYSTEM32` directory.

DELETEEVENTS
Deletes Oracle GoldenGate events from the registry.

ADDSERVICE
Adds the Manager program as a Windows service named `GGSMGR` (default) or a name specified in a GLOBALS file. Create GLOBALS as a text file (uppercase name, no file extension) in the installation directory, and specify the service name with the `MGRSERVNAME` parameter before running install.

```
MGRSERVNAME service name>
```

DELETESERVICE
Removes the Oracle GoldenGate Manager service.

AUTOSTART
If `ADDSERVICE` is used, specifies that the service starts at system boot time (the default).

MANUALSTART
If `ADDSERVICE` is used, specifies that the service starts only at user request (through GGSCI or the Services applet of the Control Panel). By default, the Manager service starts at system boot time. If `ADDSERVICE` is used, this adds the Manager program as an interactive Windows service.

USER *specification*
Specifies a user name for executing Manager. For *specification*, include the domain name, a backward slash, and the user name., For example, `HEADQT\GGSMGR`.

PASSWORD *password*
Specifies the user name password for the `USER` executing the Manager service. The password must be listed within double quotes.

WAITFORSERVICE *service name*
Specifies a service that the Manager service should wait on before starting. The server name must not contain spaces and can be obtained from the Windows Service Manager applet.

```
install.exe addservice addevents user hostname\oggmgr password "123abc"
waitforservice MSSQL$SQL2008R2
```

8.7 keygen

Use `keygen` to generate one or more encryption keys to use with Oracle GoldenGate security features that use an `ENCKEYS` file. The key values are returned to your screen. You can copy and paste them into the `ENCKEYS` file.

Syntax

```
KEYGEN key_length n
```

keygen
Used without options, the command runs the program interactively.

key_length
The length of the encryption key, up to 256 bits (32 bytes).

n
The number of keys to generate.

Syntax for IBM i CLI

```
KEYGEN [KEYLEN(key_length)] [NUMKEYS(n)]
```

KEYGEN(*key_length*)
The length of the encryption key, up to 256 bits (32 bytes).

NUMKEYS(*n*)
The number of keys to generate.

8.8 logdump

Use logdump to run the Logdump utility. This command takes no arguments and runs interactively. For more information about the Logdump utility, see *Logdump Reference for Oracle GoldenGate*.

Syntax for Windows, UNIX, and Linux

```
logdump
```

Syntax for IBM i CLI

```
LOGDUMP
```

8.9 mgr

Use `mgr` to run the Manager program from the command line of the Linux, UNIX, Windows, or IBM i operating system. The `mgr` program is installed in the Oracle GoldenGate installation directory or library.

Syntax for Windows, UNIX, and Linux

```
mgr paramfile parameter_file
[cd directory]
[pauseatend | nopauseatend]
[port portnum]
[reportfile report_file]
```

mgr
Used without options, the command runs the program interactively.

paramfile *parameter_file*
Specifies the relative or absolute path name of the parameter file for the Manager program that is being run.

cd *directory*
Changes the current working directory of the process. The process will use the specified directory for all of its operations, such as opening and writing files.

pauseatend | nopauseatend
(Windows only) When the process stops, requires an Oracle GoldenGate user to look at the console output and then strike any key to clear it. Also indicates whether the process ended normally or abnormally.

port *portnum*
The number of the first port that Manager will check to start a connection. If this port number is not available, Manager increments the number by one and continues incrementing until it finds a port number that is available. However, if a port number is specified in the Manager parameter file, that number takes precedence as the start point for this search.

reportfile *report_file*
Sends command output to the specified report file. Without the `reportfile` option, the command output is printed to the screen.

Syntax for IBM i CLI

```
MGR PARAMFILE(input_file)
[OTHERS(other_options)]
```

PARAMFILE(*input_file*)
The input text file, known as an `OBEY` file, containing the commands that you want to issue, in the order they are to be issued, one command per line. The name can be anything supported by the operating system.

OTHERS(*other_options*)
Any options that are supported in the UNIX version of the command provided as a space separated list.

8.10 replicat

Use `replicat` to run the Replicat program from the command line of the Linux, UNIX, Windows, or IBM i operating system. The `replicat` command is installed in the Oracle GoldenGate installation directory or library.

Syntax for Window, UNIX, and Linux

```
replicat paramfile parameter_file
[{atcsn CSN | aftercsn CSN} [threads(thread_list)]]
[filterduptransactions]
[initialdataload]
[pauseatend | nopauseatend]
[processid PID]
[reportfile report_file]
[skiptransaction [threads(thread_list)]]
```

replicat
Used without options, the command runs the program interactively.

paramfile *parameter_file*
Specifies the relative or absolute path name of the parameter file for the Replicat program that is being run. The default location is the `dirprm` subdirectory of the Oracle GoldenGate installation directory.

atcsn *CSN* | aftercsn *CSN* [threads(*thread_list*)]
Starts the process at or after the specified commit sequence number (CSN). F.

filterduptransactions
Causes Replicat to ignore transactions that it has already processed.

initialdataload
Runs Replicat to apply all of the data as an initial load to populate the target.

pauseatend | nopauseatend
(Windows only) When the process stops, requires an Oracle GoldenGate user to look at the console output and then strike any key to clear it. Also indicates whether the process ended normally or abnormally.

processid *PID*
A name for the process. This name must match the name that is specified for the `REPLICAT` parameter in the parameter file. Use one alphanumeric word. When used on IBM i, this name (up to the first 10 characters) will be used as the job name in the IBM i job list.

reportfile *report_file*
Sends command output to the specified report file. Without the `reportfile` option, the command output is printed to the screen. The default is the `dirrpt` subdirectory of the Oracle GoldenGate installation directory.

skiptransaction [threads(*thread_list*)]
Causes the process to skip the first transaction after its expected startup position in the trail.

Syntax for IBM i CLI

```
REPLICAT PARAMFILE(input_file)
[OTHERS(other_options)]
```

PARAMFILE(*input_file*)
The input text file, known as an `OBEY` file, containing the commands that you want to issue, in the order they are to be issued, one command per line. The name can be anything supported by the operating system.

OTHERS(*other_options*)
Any options that are supported in the UNIX version of the command provided as a space separated list.

9
Oracle GoldenGate GGSCI Commands

You can use the Oracle GoldenGate Software Command Interface (GGSCI) commands to create data replications. This is the command interface between you and Oracle GoldenGate functional components.

Topics:

- GGSCI Command Interface
  These are the GGSCI commands that you can use including examples.

9.1 GGSCI Command Interface

These are the GGSCI commands that you can use including examples.

```
Oracle GoldenGate GGSCI Commands

This file describes the commands that can be issued through the Oracle
GoldenGate
Software Command Interface (GGSCI). This is the command interface between
users and Oracle GoldenGate functional components.


Summary of Manager Commands
Use the Manager commands to control the Manager process. Manager is the parent
process of Oracle GoldenGate and is responsible for the management of its
processes
and files, resources, user interface, and the reporting of thresholds and
errors.

Command                     Description
INFO MANAGER                Returns information about the Manager port and
process id.
SEND MANAGER                Returns information about a running Manager process
and optionally
                            child processes.
START MANAGER               Starts the Manager process.
STATUS MANAGER              Returns the state of the Manager port and process
ID.
STOP MANAGER                Stops the Manager process.


Summary of Extract Commands
Use the Extract commands to create and manage Extract groups. The Extract
process
captures either full data records or transactional data changes, depending on
configuration
parameters, and then sends the data to a trail for further processing by a
downstream
process, such as a data-pump Extract or the Replicat process.

Command                     Description
ADD EXTRACT                 Creates an Extract group.
ALTER EXTRACT               Changes attributes of an Extract group
```

```
CLEANUP EXTRACT             Deletes run history for an Extract group
DELETE EXTRACT              Deletes an Extract group.
INFO EXTRACT                Returns information about an Extract group.
KILL EXTRACT                Forcibly terminates the run of an Extract group.
LAG EXTRACT                 Returns information about Extract lag.
REGISTER EXTRACT            Registers an Extract group with an Oracle database.
SEND EXTRACT                Sends instructions to, or returns information about,
a running
                            Extract group.
START EXTRACT               Starts an Extract group.
STATS EXTRACT               Returns processing statistics for an Extract group.
STATUS EXTRACT              Returns the state of an Extract group.
STOP EXTRACT                Stops an Extract group.
FORCEAPPEND                 Allows data pump to add new trail files on top of
existing initial load files
UNREGISTER EXTRACT          Unregisters an Extract group from an Oracle
database.


Summary of Replicat Commands
Use the Replicat commands to create and manage Replicat groups. The Replicat
process reads data extracted by the Extract process and applies it to target
tables
or prepares it for use by another application, such as a load application.

Command                     Description
ADD REPLICAT                Adds a Replicat group.
ALTER REPLICAT              Changes attributes of a Replicat group.
CLEANUP REPLICAT            Deletes run history of a Replicat group.
DELETE REPLICAT             Deletes a Replicat group.
INFO REPLICAT               Returns information about a Replicat group.
KILL REPLICAT               Forcibly terminates a Replicat group.
LAG REPLICAT                Returns information about Replicat lag.
REGISTER REPLICAT           Registers a Replicat group with an Oracle database.
SEND REPLICAT               Sends instructions to, or returns information about,
a running
                            Replicat group.
START REPLICAT              Starts a Replicat group.
STATS REPLICAT              Returns processing statistics for a Replicat group.
STATUS REPLICAT             Returns the state of a Replicat group.
STOP REPLICAT               Stops a Replicat group.
SYNCHRONIZE REPLICAT        Returns all threads of a coordinated Replicat to a
uniform
                            start point after an unclean shutdown of the
Replicat process.
UNREGISTER REPLICAT         Unregisters a Replicat group from an Oracle
database.


Summary of the ER Command
Use the ER command to issue standard Extract and Replicat commands to multiple
Extract and Replicat groups as a unit. See "ER" for how to use this command.

Command                     Description
INFO ER *                   Returns information about the specified wildcarded
groups.
KILL ER *                   Forcibly terminates the specified wildcarded groups.
LAG ER *                    Returns lag information about the specified
wildcarded groups
SEND ER *                   Sends instructions to, or returns information about,
the
```

```
                            specified wildcarded groups.
START ER *                  Starts the specified wildcarded groups.
STATS ER *                  Returns processing statistics for the specified
wildcarded
                            groups.
STATUS ER *                 Returns the state of the specified wildcarded groups.
STOP ER *                   Stops the specified wildcarded groups.


Summary of Wallet Commands
Use the wallet commands to manage the master-key wallet that stores Oracle
GoldenGate
master encryptions keys, and to add master keys to this wallet.

Command                     Description
CREATE WALLET               Creates a wallet that stores master encryption
keys.
OPEN WALLET                 Opens a master-key wallet.
PURGE WALLET                Permanently removes from a wallet the master keys
that are
                            marked as deleted.
ADD MASTERKEY               Adds a master key to a master-key wallet.
INFO MASTERKEY              Returns information about master keys.
RENEW MASTERKEY             Adds a new version of a master key.
DELETE MASTERKEY            Marks a master key for deletion.
UNDELETE MASTERKEY          Changes the state of a master key from being marked
as deleted
                            to marked as available.


Summary of Credential Store Commands
Use the credential store commands to manage an Oracle GoldenGate credential
store and
to add credentials to the credential store.

Command                     Description
ADD CREDENTIALSTORE         Creates a credentials store (wallet) that stores
encrypted
                            database user credentials.
ALTER CREDENTIALSTORE       Changes the contents of a credentials store.
INFO CREDENTIALSTORE        Returns information about a credentials store.
DELETE CREDENTIALSTORE      Deletes the wallet that serves as a credentials
store.


Summary of Trail Commands
Use the trail commands to create and manage Oracle GoldenGate trails. A trail is
a series
of files in which Oracle GoldenGate temporarily stores extracted data on disk
until it has
been applied to the target location.

Command                     Description

ADD EXTTRAIL                Adds a local trail to the Oracle GoldenGate
configuration.
ADD RMTTRAIL                Adds a remote trail to the Oracle GoldenGate
configuration.
ALTER EXTTRAIL              Changes attributes of a local trail.
ALTER RMTTRAIL              Changes attributes of a remote trail.
DELETE EXTTRAIL             Removes a local trail from the Oracle GoldenGate
```

```
configuration.
DELETE RMTTRAIL             Removes a remote trail from the Oracle GoldenGate
configuration.
INFO EXTTRAIL               Returns information about a local trail.
INFO RMTTRAIL               Returns information about a remote trail.


Summary of Parameter Commands
Use the parameter commands to view and manage Oracle GoldenGate parameter files.
See Administering Oracle GoldenGate for more information about how to work with
parameter
files.

Command                     Description
EDIT PARAMS                 Opens a parameter file for editing in the default
text editor.
SET EDITOR                  Sets the default text editor program for editing
parameter files.
VIEW PARAMS                 Displays the contents of a parameter file in read-
only mode on-screen.
INFO PARAM                  Returns parameter definition information.


Summary of Database Commands
Use the database commands to interact with the database from GGSCI.

Command                     Description
DBLOGIN                     Logs the GGSCI session into a database so that other
commands that
                            affect the database can be issued.
DUMPDDL                     Shows the data in the Oracle GoldenGate DDL history
table.
ENCRYPT PASSWORD            Encrypts a database login password.
FLUSH SEQUENCE              Updates an Oracle sequence so that initial redo
records are
                            available at the time that Extract starts capturing
transaction
                            data after the instantiation of the replication
environment.
LIST TABLES                 Lists the tables in the database with names that
match the input specification.
MININGDBLOGIN               Specifies the credentials of the user that an Oracle
GoldenGate process
                            uses to log into an Oracle mining database.
SET NAMECCSID               Sets the CCSID of the GGSCI session in a DB2 for i
environment.


Summary of Trandata Commands
Use trandata commands to configure the appropriate database components to
provide the
transaction information that Oracle GoldenGate needs to replicate source data
operations.

Command                     Description
ADD SCHEMATRANDATA          Enables schema-level supplemental logging.
ADD TRANDATA                Enables table-level supplemental logging.
DELETE SCHEMATRANDATA       Disables schema-level supplemental logging.
DELETE TRANDATA             Disables table-level supplemental logging.
INFO SCHEMATRANDATA         Returns information about the state of schema-level
                            supplemental logging.
```

```
INFO TRANDATA               Returns information about the state of table-level
supplemental
                            logging.
SET_INSTANTIATION_CSN       Sets whether and how table instantiation CSN
filtering is used.
CLEAR_INSTANTIATION_CSN     Clears table instantiation CSN filtering.
```

```
Summary of Checkpoint Table Commands
Use the checkpoint table commands to manage the checkpoint table that is used by
Oracle
GoldenGate to track the current position of Replicat in the trail.
For more information about checkpoints and using a checkpoint table,
see Administering Oracle GoldenGate.

Command                     Description
ADD CHECKPOINTTABLE         Creates a checkpoint table in a database.
CLEANUP CHECKPOINTTABLE     Removes checkpoint records that are no longer
needed.
DELETE CHECKPOINTTABLE      Removes a checkpoint table from a database.
INFO CHECKPOINTTABLE        Returns information about a checkpoint table.
UPGRADE CHECKPOINTTABLE     Adds a supplemental checkpoint table when upgrading
Oracle
                            GoldenGate from version 11.2.1.0.0 or earlier.
```

```
Summary of Oracle Trace Table Commands
Use the trace table commands to manage the Oracle GoldenGate trace table that is
used
with bidirectional synchronization of Oracle databases. Replicat generates an
operation
in the trace table at the start of each transaction. Extract ignores all
transactions
that begin with an operation to the trace table. Ignoring Replicat's operations
prevents data from looping back and forth between the source and target tables.
For more information about bidirectional synchronization, see Administering
Oracle GoldenGate for Windows and UNIX

Command                     Description
ADD TRACETABLE              Creates a trace table.
DELETE TRACETABLE           Removes a trace table.
INFO TRACETABLE             Returns information about a trace table.
```

```
Summary of Oracle GoldenGate Monitor JAgent Commands
Use the JAgent commands to control the Oracle GoldenGate Monitor JAgent.

Command                     Description
INFO JAGENT                 Returns information about the JAgent.
START JAGENT                Starts the JAgent.
STATUS JAGENT               Returns the state of the JAgent.
STOP JAGENT                 Stops the JAgent.
```

```
Summary of  PMSRVR COMMANDS

Use the PMSRVR commands to control the Performance Metrics Server process. The
Performance Metrics Server uses the metrics service to collect and store
instance deployment performance results.

Command                     Description
```

```
INFO PMSRVR                Returns information about the PMSRVR.
START PMSRVR               Starts the PMSRVR.
STATUS PMSRVR              Returns the state of the PMSRVR.
STOP PMSRVR                Stops the PMSRVR.
START PMSRVR               Starts the PMSRVR.
START MANAGER              Starts the Manager.
START *                    Starts Extracts and Replicats.
Start JAGENT               Starts the JAGENT
```

Summary of Oracle GoldenGate Automatic Heartbeat Commands
Use the heartbeat table commands to control the Oracle GoldenGate automatic heartbeat functionality.

```
Command                    Description
ADD HEARTBEATTABLE         Creates the objects required for automatic heartbeat
functionality.
ALTER HEARTBEATTABLE       Alters existing heartbeat objects.
DELETE HEARTBEATTABLE      Deletes existing heartbeat objects.
DELETE HEARTBEATENTRY      Deletes entries in the heartbeat table.
INFO HEARTBEATTABLE        Displays heartbeat table information.
```

Summary of Procedural Replication Commands

Use the following commands to enable, delete or retrieve information about procedures
that have supplemental logging turned on.

```
Command                    Description
ADD PROCEDURETRANDATA      Adding supplemental logging for Procedural
Replication.
DELETE PROCEDURETRANDATA   Remove supplemental logging for Procedural
Replication.
INFO PROCEDURETRANDATA     Display display supplemental logging information
about Procedural Replication.
```

Summary of Miscellaneous Oracle GoldenGate Commands
Use the following commands to control various other aspects of Oracle GoldenGate.

```
Command                    Description

!                          Executes a previous GGSCI command without
modifications.
ALLOWNESTED | NOALLOWNESTED Enables or disables the use of nested OBEY files.
CREATE SUBDIRS             Creates the default directories within the Oracle
GoldenGate home directory.
DEFAULTJOURNAL             Sets a default journal for multiple tables or files
for the ADD
                           TRANDATA command when used for a DB2 for i
database.
FC                         Allows the modification and re-execution of a
previously issued
                           Provides assistance with syntax and usage of GGSCI
commands.
HISTORY                    Shows a list of the most recently issued commands
since the
                           startup of the GGSCI session.
INFO ALL                   Displays status and lag for all Oracle GoldenGate
processes on
```

```
                            a system.
OBEY                        Processes a file that contains a list of Oracle
GoldenGate commands.
SHELL                       Executes shell commands from within the GGSCI
interface.
SHOW                        Displays the attributes of the Oracle GoldenGate
environment.
VERSIONS                    Displays information about the operating system and
database.
VIEW GGSEVT                 Displays the Oracle GoldenGate error log
(ggserr.logfile).
VIEW REPORT                 Displays the process report or the discard file that
is generated
                            by Extract or Replicat.

------------------------------------------------------------------
###################################
#
#MANAGER COMMANDS
#
#
###################################
------------------------------------------------------------------
INFO MANAGER

Use INFO MANAGER(or INFO MGR) to determine whether or not the Manager process is
running and the process ID. If Manager is running, the port number is displayed.
This
command is an alias for STATUS MANAGER.

Syntax
INFO MANAGER
INFO MGR


------------------------------------------------------------------
SEND MANAGER

Use SEND MANAGER to retrieve the status of the active Manager process or to
retrieve
dynamic port information as configured in the Manager parameter file.

Syntax
SEND MANAGER [CHILDSTATUS [DEBUG]]
[GETPORTINFO [DETAIL]
[GETPURGEOLDEXTRACTS]

CHILDSTATUS [DEBUG]

Retrieves status information about processes started by Manager. DEBUG returns
the port
numbers that are allocated to processes.

GETPORTINFO [DETAIL]

By default, retrieves the current list of ports that have been allocated to
processes
and their corresponding process IDs. DETAIL provides a list of all the ports
defined
using the DYNAMICPORTLIST parameter.
```

```
GETPURGEOLDEXTRACTS

Displays information about trail maintenance rules that are set with the
PURGEOLDEXTRACTS parameter in the Manager parameter file. For more information,
see
“PURGEOLDEXTRACTS”.

Examples

Example 1
SEND MANAGER CHILDSTATUS DEBUG returns a child process status similar to
the following. The basic CHILDSTATUS option returns the same display,
without the Port column.
ID Group Process Retry Retry Time Start Time
Port 1 ORAEXT 2400 0 None 2011/01/21 21:08:32
7840 2 ORAEXT 2245 0 None 2011/01/23 21:08:33 7842

Example 2
SEND MANAGER GETPORTINFO DETAIL returns a dynamic port list similar to the
following.
Entry Port Error Process Assigned Program
0 8000 0 2387 2011-01-01 10:30:23
1 8001 0
2 8002 0


Example 3
SEND MANAGER GETPURGEOLDEXTRACTS outputs information similar to the following.
PurgeOldExtracts Rules Fileset MinHours MaxHours MinFiles MaxFiles UseCP
S:\GGS\DIRDAT\EXTTRAIL\P4\*  0  0  1  0  Y
S:\GGS\DIRDAT\EXTTRAIL\P2\*  0  0  1  0  Y
S:\GGS\DIRDAT\EXTTRAIL\P1\*  0  0  1  0  Y
S:\GGS\DIRDAT\REPTRAIL\P4\*  0  0  1  0  Y
S:\GGS\DIRDAT\REPTRAIL\P2\*  0  0  1  0  Y
S:\GGS\DIRDAT\REPTRAIL\P1\*  0  0  1  0  Y
OK
Extract Trails
Filename  Oldest_Ch kpt_Seqno  IsTable  IsVamTwoPhaseCommit
S:\GGS\8020\DIRDAT\RT  3  0  0
S:\GGS\8020\DIRDAT\REPTRAIL\P1\RT  13  0  0
S:\GGS\8020\DIRDAT\REPTRAIL\P2\RT  13  0  0
S:\GGS\8020\DIRDAT\REPTRAIL\P4\RT  13  0  0
S:\GGS\8020\DIRDAT\EXTTRAIL\P1\ET  14  0  0
S:\GGS\8020\DIRDAT\EXTTRAIL\P2\ET  14  0  0
S:\GGS\8020\DIRDAT\EXTTRAIL\P4\ET  14  0  0


------------------------------------------------------------------------
START MANAGER

Use START MANAGER to start the Manager process. This applies to a non-clustered
environment. In a Windows cluster, you should stop Manager from the Cluster
Administrator.

Syntax
START MANAGER


------------------------------------------------------------------------
STATUS MANAGER
```

```
Use STATUS MANAGER to see if the Manager process is running and any associate
process ID. If Manager is running, the port number is displayed.

Syntax
STATUS MANAGER

------------------------------------------------------------------------
STOP MANAGER

Use STOP MANAGER to stop the Manager process. This applies to non-clustered
environments. In a Windows cluster, Manager must be stopped through the
Cluster Administrator.

Syntax
STOP MANAGER [!]

! (Exclamation point) Bypasses the prompt that confirms the intent to shut
down Manager.

Examples:

STOP MANAGER

STOP MANAGER !

------------------------------------------------------------------------
####################################
#
#EXTRACT COMMANDS
#
#
####################################

------------------------------------------------------------------------
ADD EXTRACT

Use ADD EXTRACT to create an Extract group. Unless a SOURCEISTABLE task or
an alias Extract is specified, ADD EXTRACT creates an online group that uses
checkpoints so that processing continuity is maintained from run to run.

For DB2 for i, this command establishes a overall start point for all journals
and is a required first step. After issuing the ADD EXTRACT command, you can
then optionally position any given journal at a specific journal sequence
number by using the ALTER EXTRACT command with an appropriate journal option.

Oracle GoldenGate supports up to 5,000 concurrent Extract and Replicat
groups per instance of Oracle GoldenGate Manager. At the supported level,
all groups can be controlled and viewed in full with GGSCI commands such
as the INFO and STATUS commands. Oracle GoldenGate recommends keeping the
combined number of Extract and Replicat groups at the default level of
300 or below in order to manage your environment effectively.

This command cannot exceed 500 bytes in size for all keywords and input,
including any text that you enter for the DESC option.

Syntax for a Regular, Passive, or Data Pump Extract

ADD EXTRACT group_name
{, SOURCEISTABLE |
   , TRANLOG [bsds_name  |
   , INTEGRATED TRANLOG |
```

```
   , VAM |
   , EXTFILESOURCE file_name |
   , EXTTRAILSOURCE trail_name |
   , VAMTRAILSOURCE VAM_trail_name}
 BEGIN {NOW | yyyy-mm-dd[ hh:mi:[ss[.cccccc]]]} |
 EXTSEQNO sequence_number, EXTRBA relative_byte_address |
 EOF |
 LRI (value)|
 LSN (value) |
 EXTRBA relative_byte_address |
 PAGE data_page, ROW row_ID |
 SEQNO sequence_number
 SCN value
 THREADS [n]
 PASSIVE
 PARAMS file_name
 REPORT file_name
 DESC 'description'
 SOCKSPROXY {host_name | IP_address}[:port] [PROXYCSALIAS credential_store_alias
[PROXYCSDOMAIN credential_store_domain]]]
 RMTNAME passive_Extract_name]
 DESC [description]
}

group_name

The name of the Extract group. The name of an Extract group can contain
up to eight characters. See for group naming conventions. See Administering
Oracle GoldenGate for group naming conventions.

SOURCEISTABLE

Creates an Extract task that extracts entire records from the database
for an initial load using the Oracle GoldenGate direct load method or
the direct bulk load to SQL*Loader method. If SOURCEISTABLE is not
specified, ADD EXTRACT creates an online change-synchronization process,
and one of the other data source options must be specified. When using
SOURCEISTABLE, do not specify any service options. Task parameters
must be specified in the parameter file.

For more information about initial load methods, see Administering Oracle
GoldenGate.

TRANLOG [bsds_name]

Specifies the transaction log as the data source. Use this option for all
databases. TRANLOG requires the BEGIN option.

(DB2 on z/OS) You can use the bsds_name option for DB2 on a z/OS system to
specify the Bootstrap Data Set file name of the transaction log, though it
is not required and is not used. You do not need to change existing TRANLOG
parameters.

(DB2 LUW) You can use the LRI option for DB2 LUW systems to specify
the LRI at which Extract can start capturing records from the transaction
log. You can use the DB2 utility db2logsForRfwd to obtain the LRI. This
utility provides LRI ranges present in the DB2 logs. Note that,
although Extract might position to a given LRI, that LRI might not
necessarily be the first one that Extract processes. There are
numerous record types in the log files that Extract ignores, such as
DB2 internal log records. Extract reports the actual starting LRI
```

to the Extract report file.

(Oracle) As of Oracle Standard or Enterprise Edition 11.2.0.3, this mode is known as classic capture mode. Extract reads the Oracle redo logs directly. See INTEGRATED TRANLOG for an alternate configuration.

INTEGRATED TRANLOG

(Oracle) Adds this Extract in integrated capture mode. In this mode, Extract integrates with the database logmining server, which passes logical change records (LCRs) directly to Extract. Extract does not read the redo log. Before using INTEGRATED TRANLOG, use the REGISTER EXTRACT command. For information about integrated capture, see the Oracle GoldenGate documentation for your database.

VAM

(MySQL and Teradata) Specifies that the Extract API known as the Vendor Access Module (VAM) will be used to transfer change data to Extract.

EXTFILESOURCE file_name

Specifies an extract file as the data source. Use this option with a secondary Extract group (data pump) that acts as an intermediary between a primary Extract group and the target system.

For file_name, specify the relative or fully qualified path name of the file, for example dirdat/extfile or c:\ggs\dirdat\extfile.

EXTTRAILSOURCE trail_name

Specifies a trail as the data source. Use this option with a secondary Extract group (data pump) that acts as an intermediary between a primary Extract group and the target system.

For trail_name, specify the relative or fully qualified path name of the trail, for example dirdat/aa or c:\ggs\dirdat\aa.

BEGIN {NOW | yyyy-mm-dd[hh:mi:[ss[.ccccc]]]}
Specifies a timestamp in the data source at which to begin processing.

NOW

For all databases except DB2 LUW, NOW specifies the time at which the ADD EXTRACT command is issued.

For DB2 LUW, NOW specifies the time at which START EXTRACT takes effect. It positions to the first record that approximately matches the date and time. This is because the only log records that contain timestamps are the commit and stop transaction records, so the starting position can only be calculated relative to those timestamps. This is a limitation of the API that is used by Oracle GoldenGate. It must be noted that positioning by timestamp is not accurate and can also take a long time. It is recommended to use LRI or EOF options wherever possible.

Do not use NOW for a data pump Extract except to bypass data that was captured to the trail prior to the ADD EXTRACT statement.

```
    yyyy-mm-dd[ hh:mi:[ss[.cccccc]]]

    A date and time (timestamp) in the given form. For an Oracle Extract in
    integrated mode, the timestamp value must be greater than the timestamp at
    which the Extract was registered with the database.

    * Positioning by timestamp in a SQL Server transaction log is affected by
    the following characteristics of SQL Server:

      – The timestamps recorded in the SQL Server transaction log use a 3.333
      microsecond (ms) granularity. This level of granularity may not allow
      positioning by time between two transactions, if the transactions began
      in the same 3.333 ms time interval.

      – Timestamps are not recorded in every SQL Server log record, but only in
      the records that begin and commit the transaction, as well as some others
      that do not contain data.

      – SQL Server timestamps are not from the system clock, but instead are
      from an internal clock that is specific to the individual processors in
use.
      This clock updates several times a second, but between updates it could
get
      out of sync with the system clock. This further reduces the precision of
      positioning by time.

      – Timestamps recorded for log backup files may not precisely correspond to
      times recorded inside the backup (however this imprecision is less than a
      second).

    Positioning to an LSN is precise.

EXTSEQNO sequence_number, EXTRBA relative_byte_address

Valid for a primary Extract in classic capture mode for Oracle and a data pump
Extract. Not supported for an Oracle Extract in integrated mode. Specifies
either of the following:

* sequence number of an Oracle redo log and RBA within that log at which
to begin capturing data.

* the file in a trail in which to begin capturing data (for a data pump).
Specify the sequence number, but not any zeroes used for padding. For
example, if the trail file is c:\ggs\dirdat\aa000000026, you would specify
EXTSEQNO 26. By default, processing begins at the beginning of a trail
unless this option is used.

Contact Oracle Support before using this option. For more information, go to
http://support.oracle.com.

EXTRBA relative_byte_address

Valid for DB2 on z/OS. Specifies the relative byte address within a transaction
log at which to begin capturing data. The required format is 0Xnnn, where
nnn is a 1 to 20 digit hexadecimal number (the first character is the digit
zero, and the second character can be upper or lower case letter x).

EOF

Valid for SQL Server, DB2 LUW and DB2 for i. Configures processing to start at
```

the end of the log files (or journals) that the next record will be written to. Any active transactions will not be captured.

LRi (value)
Valid for DB2 LUW. Specifies a start position in the transaction logs when Extract starts.

You can use the LRI option for DB2 LUW systems to specify the LRI at which Extract can start capturing records from the transaction log. You can use the DB2 utility db2logsForRfwd to obtain the LRI. This utility provides LRI ranges present in the DB2 logs.

Note that, although Extract might position to a given LRI, that LRI might not necessarily be the first one that Extract will process. There are numerous record types in the log files that Extract ignores, such as DB2 internal log records. Extract will report the actual starting LRI to the Extract report file.

LSN [value]

Valid for SQL Server. Specifies the LSN in a transaction log at which to start capturing data. The specified LSN should exist in a log backup or the online log. An alias for this option is EXTLSN.

For SQL Server, an LSN is composed of one of these, depending on how the database returns it:

* Colon separated hex string (8:8:4) padded with leading zeroes and 0X prefix, as in 0X00000d7e:0000036b:01bd

* Colon separated decimal string (10:10:5) padded with leading zeroes, as in 0000003454:0000000875:00445

* Colon separated hex string with 0X prefix and without leading zeroes, as in 0Xd7e:36b:1bd

* Colon separated decimal string without leading zeroes, as in 3454:875:445

* Decimal string, as in 3454000000087500445

In the preceding, the first value is the virtual log file number, the second is the segment number within the virtual log, and the third is the entry number. You can find the LSN for named transactions by using a query like:
select [Current LSN], [Transaction Name], [Begin Time] from fn_dblog(null, null) where Operation = 'LOP_BEGIN_XACT' and [Begin Time] = 'time'

The time format that you should use in the query should be similar to '2015/01/30 12:00:00.000' and not '2017-01-30 12:00:00.000'.

You can determine the time that a particular transaction started, then find the relevant LSN, and then position between two transactions with the same begin time.

SEQNO sequence_number

Valid for DB2 for i. Starts capture at, or just after, a system sequence number, which is a decimal number up to 20 digits in length.

```
SCN [value]

Valid for Oracle. Starts Extract at the transaction in the redo log
that has the specified Oracle system change number (SCN). This option
is valid for Extract both in classic capture and integrated modes. For
Extract in integrated mode, the SCN value must be greater than the SCN
at which the Extract was registered with the database. For more
information, see REGISTER EXTRACT.

PARAMS file_name

Specifies the full path name of an Extract parameter file in a location
other than the default of dirprm within the Oracle GoldenGate directory.

REPORT file_name

Specifies the full path name of an Extract report file in a location other
than the default of dirrpt within the Oracle GoldenGate directory.

THREADS [n]

Valid for Oracle classic capture mode. Specifies the number of producer
threads that Extract maintains to read redo logs.

Required in an Oracle RAC configuration to specify the number of producer
threads. These are the Extract threads that read the different redo logs
on the various RAC nodes. The value must be the same as the number of nodes
from which you want to capture redo data.


PASSIVE

Specifies that this Extract group runs in passive mode and can only be
started and stopped by starting or stopping an alias Extract group on
the target system. Source-target connections will be established not
by this group, but by the alias Extract from the target.

This option can be used for a regular Extract group or a data-pump
Extract group. It should only be used by whichever Extract on the
source system is the one that will be sending the data across the
network to a remote trail on the target.

For instructions on how to configure passive and alias Extract groups,
see Administering Oracle GoldenGate.

DESC 'description'

Specifies a description of the group, such as 'Extracts account_tab on
Serv1'. Enclose the description within single quotes. You may use the
abbreviated keyword DESC or the full word DESCRIPTION.

SOCKSPROXY{host_name| IP_address}[:port] [PROXYCSALIAS credential_store_alias
[PROXYCSDOMAINcredential_store_domain]

Use for an alias Extract. Specifies the DNS host name or IP address of the
proxy server. You can use either one to define the host though you must use
the IP address if your DNS server is unreachable. If you are using an IP
address, use either an IPv6 or IPv4 mapped address, depending on the stack
of the destination system. You must specify the PROXYCSALIAS. In addition,
you can specify the port to use, and the credential store domain.
```

```
RMTNAME passive_extract_name

Use for an alias Extract. Specifies the passive Extract name, if different
from that of the alias Extract.

Examples

Example 1

The following creates an Extract group named finance that extracts database
changes from the transaction logs. Extraction starts with records generated
at the time when the group was created with ADD EXTRACT.

ADD EXTRACT finance, TRANLOG, BEGIN NOW

Example 2

The following creates an Extract group named finance that extracts database
changes from Oracle RAC logs. Extraction starts with records generated at
the time when the group was created. There are four RAC instances, meaning
there will be four Extract threads.

ADD EXTRACT finance, TRANLOG, BEGIN NOW, THREADS 4

Example 3

The following creates an Extract group named finance that extracts database
changes from the transaction logs. Extraction starts with records generated
at 8:00 on January 21, 2011.

ADD EXTRACT finance, TRANLOG, BEGIN 2017-01-21 08:00

Example 4

The following creates an integrated capture Extract group.
ADD EXTRACT finance, INTEGRATED TRANLOG, BEGIN NOW

Example 5

The following creates an Extract group named finance that interfaces with
a Teradata TAM in either maximum performance or maximum protection mode.
No BEGIN point is used for Teradata sources.

ADD EXTRACT finance, VAM

Example 6

The following creates a data-pump Extract group named finance. It reads
from the Oracle GoldenGate trail c:\ggs\dirdat\lt.

ADD EXTRACT finance, EXTTRAILSOURCE dirdat/lt

Example 7

The following creates an initial-load Extract named load.

ADD EXTRACT load, SOURCEISTABLE

Example 8

The following creates a passive Extract group named finance that extracts
```

```
database changes from the transaction logs.

ADD EXTRACT finance, TRANLOG, BEGIN NOW, PASSIVE

Example 9

The following creates an alias Extract group named financeA. The alias
Extract is associated with a passive extract named finance on source system
sysA. The Manager on that system is using port 7800.

ADD EXTRACT financeA, RMTHOST sysA, MGRPORT 7800, RMTNAME finance

Example 10

The following examples create and position Extract at a specific Oracle
system change number (SCN) in the redo log.

ADD EXTRACT finance TRANLOG SCN 123456
ADD EXTRACT finance INTEGRATED TRANLOG SCN 123456


Example 11

The following example creates an alias Extract specifying the host to use.

ADD EXTRACT apmp desc "alias extract" RMTHOST lc01abc MGRPORT 7813 RMTNAME
ppmp SOCKSPROXY lc02def:3128 PROXYCSALIAS proxyAlias

Example 12

The following example creates an Extract on a DB2 LUW system.

ADD EXTRACT extcust, TRANLOG LRI 8066.322711

------------------------------------------------------------------------
ALTER EXTRACT

Use ALTER EXTRACT for the following purposes:

* To change the attributes of an Extract group created with the ADD
EXTRACT command.

* To increment a trail to the next file in the sequence.

* To upgrade to an integrated capture configuration.

* To downgrade from an integrated capture configuration.

* To position any given IBM for i journal at a specific journal
sequence number.

Before using this command, stop Extract with the STOP EXTRACT
group_name command.

Syntax

ALTER EXTRACT group_name
 [ADD_EXTRACT_attribute]
 [TRANLOG LRI_number]
 [UPGRADE INTEGRATED TRANLOG]
 [DOWNGRADE INTEGRATED TRANLOG [THREADS number]]
```

```
 [THREAD number]
 [SCN value]
 [ETROLLOVER]

The following ALTER EXTRACT options are supported for DB2 for i
to position Extract for a given journal:

ALTER EXTRACT {BEGIN {NOW | yyyy-mm-dd[ hh:mi:[ss[.cccccc]]]}
[JOURNAL journal_library/journal_name [JRNRCV receiver_library/
receiver_name]] | , EOF [JOURNAL journal_library/journal_name
[JRNRCV receiver_library/receiver_name]] | , SEQNO sequence_number
[JOURNAL journal_library/journal_name [JRNRCV receiver_library/
receiver_name]]}

group_name

The name of the Extract group that is to be altered.

ADD_EXTRACT_attribute

You can change any of the attributes specified with the ADD EXTRACT
command, except for the following:

* Altering an Extract specified with the EXTTRAILSOURCE option.

* Altering the number of RAC threads specified with the THREADS option.


For these exceptions, delete the Extract group and then add it again.

If using the BEGIN option, do not combine other options in the statement.
Issue separate statements, for example:

ALTER EXTRACT finance, BEGIN 2017-01-01
ALTER EXTRACT finance, ETROLLOVER
ALTER EXTRACT finance, SCN 789000

If using the SCN or BEGIN option for Integrated Extract, it requires a DBLOGIN,
and the SCN or timestamp value specified cannot be below the outbound
server's first SCN or timestamp. To find the outbound server's first SCN,
issue the following command:

INFO EXTRACT group_name, SHOWCH DETAIL

The first SCN value is listed as shown in the following example:

Integrated Extract outbound server first scn: 0.665884 (665884)

TRANLOG LRI_number
(DB2 LUW) You can use the LRI_number option for DB2 LUW systems to
specify the LRI record value for the checkpoint transaction log.

UPGRADE INTEGRATED TRANLOG

Upgrades the Extract group from classic capture to integrated capture.
To support the upgrade, the transaction log that contains the start of
the oldest open transaction must be available on the source or downstream
mining system. For instructions on making the transition from classic to
integrated capture, see the full procedure in Administering Oracle
GoldenGate.
```

```
DOWNGRADE INTEGRATED TRANLOG [THREADS number]

Downgrades the Extract group from integrated capture to classic capture.
When downgrading on a RAC system, the THREADS option must be used to specify
the number of RAC threads. On a non-RAC system, you can optionally specify
THREADS 1 to cause the downgraded classic Extract to run in threaded mode
with one thread, which is similar to doing an ADD EXTRACT with THREADS 1
on a non-RAC system.

See Administering Oracle GoldenGate for the full procedure for performing the
transition from integrated to classic capture.

To support the downgrade, the transaction log that contains the start
of the oldest open transaction must be available on the source or
downstream mining system. For information about integrated capture,
see the Oracle GoldenGate documentation for your database.

THREAD number

Valid for classic capture mode. In an Oracle RAC configuration,
alters Extract only for the specified redo thread. Only one thread
number can be specified.


SCN value

Valid for Oracle. Repositions Extract to the transaction in the
redo log that has the specified Oracle system change number (SCN). You
cannot alter the Extract to an SCN less than the first SCN.
This option is valid both for integrated capture mode and classic
capture mode.

ETROLLOVER

Use for manual recovery situations that require repositioning and
regenerating trail files for a primary Extract and when upgrading Oracle
GoldenGate from a previous version.
Causes Extract to create a new incarnation of the trail file and increments
to the next file in the trail sequence when restarting, requiring readers
such as Pump or Replicat, to be manually repositioned to the new trail
sequence number.

For example, if the current file is ET000000002, the current file will be
ET000000003 when Extract restarts. A trail can be incremented from 000000001
through 999999999, and then the sequence numbering starts over at 000000000.


BEGIN {NOW | yyyy-mm-dd[ hh:mi:[ss[.cccccc]]]}
[JOURNALjournal_library/journal_name
[JRNRCV receiver_library/ receiver_name]] |
, EOF [JOURNALjournal_library/journal_name
[JRNRCVreceiver_library/receiver_name]] |
, SEQNO sequence_number[JOURNALjournal_library/journal_name
[JRNRCV receiver_library/receiver_name]]

These IBM for i options allow journal-specific Extract positioning after
the extract overall position is issued with ADD EXTRACT. A specific journal
position set with ALTER EXTRACT does not affect any overall position that
was previously set with ADD EXTRACT or ALTER EXTRACT; however an extract
overall position set with ALTER EXTRACT overrides any specific journal positions
that were previously set in the same Extract configuration.
```

Note:
SEQNO, when used with a journal in ALTER EXTRACT, is the journal sequence number that is relative to that specific journal, not the system sequence number that is common across all of the journals read by the Extract.

Examples

Example 1

The following alters Extract to start processing data from January 1, 2011.

```
ALTER EXTRACT finance, BEGIN 2017-01-01
```

Example 2

The following alters Extract to start processing at a specific location in the trail.

```
ALTER EXTRACT finance, EXTSEQNO 26, EXTRBA 338
```

Example 3

The following alters Extract in an Oracle RAC environment, and applies the new begin point only for redo thread 4.

```
ALTER EXTRACT accounts, THREAD 4, BEGIN 2017-01-01
```

Example 4

The following alters Extract to increment to the next file in the trail sequence.

```
ALTER EXTRACT finance, ETROLLOVER
```

Example 5

The following alters Extract to upgrade to integrated capture.

```
ALTER EXTRACT finance, UPGRADE INTEGRATED TRANLOG
```

Example 6

The following alters Extract to downgrade to classic capture in a RAC environment.

```
ALTER EXTRACT finance, DOWNGRADE INTEGRATED TRANLOG THREADS 3
```

Example 7

The following alters Extract in an Oracle environment to start processing data from source database SCN 778899.

```
ALTER EXTRACT finance, SCN 778899
```

Example 8

The following shows ALTER EXTRACT for an IBM for i journal start point.

```
ALTER EXTRACT finance, SEQNO 1234 JOURNAL accts/acctsjrn
```

```
Example 9

The following shows ALTER EXTRACT for an IBM for i journal and
receiver start point.

ALTER EXTRACT finance, SEQNO 1234 JOURNAL accts/acctsjrn JRNRCV
accts/jrnrcv0005

Example 10

The following example alters an Extract on a DB2 LUW system.

ALTER EXTRACT extcust, TRANLOG LRI 8066.322711

------------------------------------------------------------------------
CLEANUP EXTRACT

Use CLEANUP EXTRACT to delete run history for the specified Extract
group. The cleanup keeps the last run record intact so that Extract
can resume processing from where it left off. Before using this command,
stop Extract by issuing the STOP EXTRACT command.

Syntax
CLEANUP EXTRACT group_name  SAVE count]

group_name
The name of an Extract group or a wildcard (*) to specify multiple groups.
For example, T* cleans up all Extract groups whose names start with T.

SAVE count
Excludes the specified number of the most recent records from the cleanup.

Examples

Example 1
The following deletes all but the last record.

CLEANUP EXTRACT finance

Example 2
The following deletes all but the most recent five records.

CLEANUP EXTRACT *, SAVE 5

------------------------------------------------------------------------
DELETE EXTRACT

Use DELETE EXTRACT to delete an Extract group. This command deletes the
checkpoint file that belongs to the group, but leaves the parameter file
intact. You can then re-create the group or delete the parameter file as needed.

Before using DELETE EXTRACT, stop Extract with the STOP EXTRACT command.

To delete the trail files that are associated with the Extract group,
delete them manually through the operating system.

Syntax
DELETE EXTRACT group_name [!]

group_name
```

```
The name of an Extract group or a wildcard specification (*) to specify
multiple groups. For example, T* deletes all Extract groups whose names
start with T.

!
(Exclamation point) Deletes all Extract groups associated with a wildcard
without prompting.

------------------------------------------------------------------------
INFO EXTRACT

Use INFO EXTRACT to view the following information.

* The status of Extract (STARTING, RUNNING, STOPPED, or ABENDED). STARTING
means that the process has started but has not yet locked the checkpoint
file for processing.

* Approximate Extract lag.

* Checkpoint information.

* Process run history.

* The trail(s) to which Extract is writing.

* Status of upgrade to, or downgrade from, Integrated Extract

  Extract can be running or stopped when INFO EXTRACT is issued. In the case
  of a running process, the status of RUNNING can mean one of the following:

        * Active: Running and processing (or able to process) data. This is the
          normal state of a process after it is started.

        * Suspended: The process is running, but suspended due to an
EVENTACTIONS
          SUSPEND action. In a suspended state, the process is not active, and
no data
          can be processed, but the state of the current run is preserved and
can be
          continued by issuing the SEND EXTRACT command with the RESUME option
in GGSCI.
          The RBA in the INFO command reflects the last checkpointed position
before
          the suspend action. To determine whether the state is active or
suspended,
          issue the SEND EXTRACT command with the STATUS option.


The basic command displays information only for online (continuous) Extract
processes. Tasks are excluded.

About Extract Lag

The Checkpoint Lag field of the INFO EXTRACT output reflects the lag, in
seconds,
at the time that the last checkpoint was written to the trail. For example,
if the following is true...

* Current time = 15:00:00
```

```
* Last checkpoint = 14:59:00

* Timestamp of the last record processed = 14:58:00


...then the lag is reported as 00:01:00 (one minute, the difference between
14:58
and 14:59).

A lag value of UNKNOWN indicates that the process could be running but has not
yet processed records, or that the source system's clock is ahead of the target
system's clock (due to clock imperfections, not time zone differences).

For more precise lag information, use LAG EXTRACT (see “LAG EXTRACT”).

Syntax

INFO EXTRACT group_name
 [SHOWCH [n]]
 [DETAIL]
 [TASKS | ALLPROCESSES]
 [UPGRADE | DOWNGRADE | CONTAINERS]

group_name

The name of an Extract group or a wildcard (*) to specify multiple groups.
For example, T* shows information for all Extract groups whose names start
with T.

SHOWCH [n]

The basic command shows information about the current Extract checkpoints.
Extract checkpoint positions are composed of read checkpoints in the data
source and write checkpoints in the trail. The trail type (RMTTRAIL or
EXTTRAIL) is also noted.

Optionally, specify a value for n to include the specified number of previous
checkpoints as well as the current one.

Note:
You might see irregular indents and spacing in the output. This is normal
and does not affect the accuracy of the information. See Administering
Oracle GoldenGate for more information about checkpoints, including descriptions
of the types of checkpoints made by each process and the internal metadata
entries
that are included in the display.

DETAIL

Displays the following:

* Extract run history, including start and stop points in the data source,
expressed as a time.

* Trails to which Extract is writing.

See Example 5 for sample output of DETAIL.

TASKS

Displays only Extract tasks. Tasks that were specified by a wildcard
```

```
argument are not displayed by INFO EXTRACT.

ALLPROCESSES

Displays all Extract groups, including tasks.

UPGRADE | DOWNGRADE

Valid for an Oracle database only.

* UPGRADE displays whether the Extract can be upgraded from classic capture
mode to integrated capture mode.

* DOWNGRADE displays whether the Extract can be downgraded from integrated
capture mode to classic capture mode.


If Extract cannot be upgraded or downgraded, the reason is displayed.

A wildcarded Extract name is not allowed with this option.

Before using this command, issue the DBLOGIN command.

CONTAINERS

Lists the PDBs that are registered with the specified Extract group.
However, the command errors out if it is run in non-CDB mode or the
Extract group doesn't exist.

Issue the DBLOGIN command before running this command.

Examples

Example 1

INFO EXTRACT fin*, SHOWCH

Example 2

INFO EXTRACT *, TASKS

Example 3 (Oracle only)

INFO EXTRACT finance UPGRADE

Example 4

The following example shows basic INFO EXTRACT output.

EXTRACT EXTCUST Last Started 2017-01-05 16:09 Status RUNNING
Checkpoint Lag             00:01:30 (updated 97:16:45 ago)
Log Read Checkpoint File     /rdbms/data/oradata/redo03a.log
               2017-01-05 16:05:17 Seqno 2952, RBA 7598080
Example 5

The following is an example of the output of INFO EXTRACT with DETAIL.
EXTRACT ORAEXT             Last Started 2017-01-15 16:16 Status STOPPED
Checkpoint Lag             00:00:00 (updated 114:24:48 ago)
Log Read Checkpoint         File C:\ORACLE\ORADATA\ORA920\REDO03.LOG
               2017-01-15 16:17:53 Seqno 46, RBA 3757568
Target Extract Trails:
```

```
Trail Name             Seqno  RBA  Max MB  Trail Type

 c:\goldengate802\dirdat\xx      0  57465  10  RMTTRAIL
 c:\goldengate802\dirdat\jm      0  19155  10  RMTTRAIL

Extract Source  Begin  End

C:\ORACLE\ORADATA\ORA920\REDO03.LOG  2017-01-15 16:07 2017-01-15 16:17
C:\ORACLE\ORADATA\ORA920\REDO03.LOG  2017-01-15 15:55  2017-01-15 16:07
C:\ORACLE\ORADATA\ORA920\REDO03.LOG  2017-01-15 15:42  2017-01-15 15:55
C:\ORACLE\ORADATA\ORA920\REDO03.LOG  2017-01-15 15:42  2017-01-15 15:42
 Not Available  * Initialized *  2017-01-15 15:42

Current directory     C:\GoldenGate802
Report file         C:\GoldenGate802\dirrpt\ORAEXT.rpt
Parameter file         C:\GoldenGate802\dirprm\ORAEXT.prm
Checkpoint file     C:\GoldenGate802\dirchk\ORAEXT.cpe
Process file         C:\GoldenGate802\dirpcs\ORAEXT.pce
Error log         C:\GoldenGate802\ggserr.log

------------------------------------------------------------------------
KILL EXTRACT

Use KILL EXTRACT to kill an Extract process running in regular or PASSIVE mode.
Use this command only if a process cannot be stopped gracefully with the STOP
EXTRACT command. The Manager process will not attempt to restart a killed
Extract process.

Syntax

KILL EXTRACT group_name

group_name

The name of an Extract group or a wildcard (*) to specify multiple groups.
For example, T* kills all Extract processes whose group names start with T.

Example

KILL EXTRACT finance

------------------------------------------------------------------------
LAG EXTRACT

Use LAG EXTRACT to determine a true lag time between Extract and the data
source. LAG EXTRACT calculates the lag time more precisely than INFO EXTRACT
because it communicates with Extract directly, rather than reading a
checkpoint position in the trail.

For Extract, lag is the difference, in seconds, between the time that a record
was processed by Extract (based on the system clock) and the timestamp of
that record in the data source.

If the heartbeat functionality is enabled, you can view the associated lags.

Syntax
LAG EXTRACT
 [group_name]
 [GLOBAL]
```

```
group_name

The name of an Extract group or a wildcard (*) to specify multiple groups.
For example, T* determines lag time for all Extract groups whose names
start with T.

GLOBAL

Displays the lags in the GG_LAGS view.

Examples

Example 1

LAG EXTRACT *


Example 2

LAG EXTRACT *fin*


Example 3

The following is sample output for LAG EXTRACT.

Sending GETLAG request to EXTRACT CAPTPCC...
Last record lag: 2 seconds.
At EOF, no more records to process.

------------------------------------------------------------------------
REGISTER EXTRACT

Use REGISTER EXTRACT to register a primary Extract group with an Oracle
Database to:

* Enable integrated capture mode

* Specify options for Integrated Extract from a multitenant container database

* Enable Extract in classic capture mode to work with Oracle Recovery Manager
to retain the archive logs needed for recovery


REGISTER EXTRACT is not valid for a data pump Extract.

To unregister an Extract group from the database, use the UNREGISTER EXTRACT
command (see “UNREGISTER EXTRACT”). See the Oracle GoldenGate documentation for
your database
for more information about using REGISTER EXTRACT.

Syntax

For classic Extract:

REGISTER EXTRACT group_name LOGRETENTION

For Integrated Extract:

REGISTER EXTRACT group-name
  ( LOGRETENTION | DATABASE
```

```
     ( [ CONTAINER container-list |
         ADD CONTANER container-list |
         DROP CONTAINER container-list ]
       [ SCN scn ]
       [ SHARE ( AUTOMATIC | group-name | NONE ) ]
       [ [NO]OPTIMIZED ]
)
)
Container-list is a comma separated list of PDB names, for example (pdb1, pdb2);
or wildcarded PDB names, for example (pdb* or pdb?); or both, for example
(cdb1_pdb1, pdb*). Supported wildcards are ? and *.
The default value is NOOPTIMIZED. The NOOPTIMIZED option cannot be used with
the ADD CONTAINER or DROP CONTAINERor SHARE EXTRACT options.

group_name

The name of the Extract group that is to be registered. Do not use a wildcard.
DATABASE[
CONTAINER (container ...]) |
ADD CONTAINER (container ...]) |
DROP CONTAINER (container ...])
]

Without options, DATABASE enables integrated capture from a non-CDB database
for the Extract group. In this mode, Extract integrates with the database
logmining server to receive change data in the form of logical change records
(LCR). Extract does not read the redo logs. Extract performs capture processing,
transformation, and other requirements. The DML filtering is performed by the
Logmining server. For support information and configuration steps, see
the Oracle GoldenGate documentation for your database.

Before using REGISTER EXTRACT with DATABASE, use the DBLOGIN command for all
extracts with the privileges granted using the
dbms_goldengate_auth.grant_admin_privilege
procedure. If you have a downstream configuration, then you must also issue the
MININGDBLOGIN command. If the source database you are registering is a CDB
database
and Extract will fetch data, then grant_admin_privilege must be called with the
CONTAINER='ALL' parameter.

After using REGISTER EXTRACT, use ADD EXTRACT with the INTEGRATED TRANLOG option
to create an Extract group of the same name. You must register an Extract group
before adding it.

    CONTAINER (container ...])
    (containers) of a multitenant container database (CDB). Specify one or more
    pluggable databases as a comma-delimited list within parentheses, for
    example: CONTAINER (pdb1, pdb2, pdb3). If you list the pluggable
    databases, they must exist in the database. You can also specify the
    pluggable databases using the wildcards * and ?. For example, CONTAINER
    (pdb*).

    ADD CONTAINER (container ...])
    Adds the specified pluggable database to an existing Extract capture
configuration.
    Specify one or more pluggable databases as a comma-delimited list within
    parentheses, or using the wildcards * and ?. For example: ADD CONTAINER
(pdb1, pdb2,
    pdb3). Before issuing     REGISTER EXTRACT with this option, stop the
Extract group.
```

```
    For Oracle, adding CONTAINERs at particular SCN on an existing Extract
    is not supported.

    DROP CONTAINER (container ...])
    Drops the specified pluggable database from an existing Extract capture
    configuration. Specify one or more pluggable databases as a comma-delimited
    list within parentheses, for example: DROP CONTAINER (pdb1, pdb2, pdb3).
    A register drop container does not fully happen until the Extract has been
    started and it reads a committed txn from a dropped pluggable database
greater
    than the Extract checkpoint SCN. Extract then fully drops the containers
then
    shutdowns with a message.
    Before issuing REGISTER EXTRACT with this option, stop the Extract group.

LOGRETENTION

Valid for classic Extract only. Enables an Extract group in classic capture mode
to work with Oracle Recovery Manager (RMAN) to retain the logs that Extract
needs
for recovery. LOGRETENTION is ignored if the Extract group is configured for
integrated capture.

LOGRETENTION creates an underlying Oracle Streams capture process that is
dedicated
to the Extract group and has a similar name. This capture is used only for the
purpose of log retention.

The logs are retained from the time that REGISTER EXTRACT is issued, based on
the
current database SCN. The log-retention feature is controlled with the
LOGRETENTION
option of the TRANLOGOPTIONS parameter.

Before using REGISTER EXTRACT with LOGRETENTION, issue the DBLOGIN command with
the privileges shown in “DBLOGIN”.

SCN scn

Registers Extract to begin capture at a specific system change number (SCN)
in the past. Without this option, capture begins from the time that REGISTER
EXTRACT
is issued. The specified SCN must correspond to the begin SCN of a dictionary
build operation in a log file. You can issue the following query to find all
valid SCN values:

SELECT first_change#
  FROM v$archived_log
WHERE dictionary_begin = 'YES' AND
  standby_dest = 'NO' AND
  name IS NOT NULL AND
  status = 'A';

When used alone, the SCN value is the beginning SCN of the dictionary build
operation in a log file.

When used in conjunction with SHARE AUTOMATIC or SHARE extract_name, then the
specified SCN is the start_scn for the capture session and has the following
restrictions:

* Should be lesser than or equal to the current SCN.
```

```
* Should be greater than the minimum (first SCN) of the existing captures.


{SHARE [
AUTOMATIC |

extract|
NONE]}

Registers the extract to return to an existing LogMiner data dictionary build
with a specified SCN creating a clone. This allows for faster creation captures
by leveraging existing dictionary builds.

SHARE cannot be used on a CDB.

The following GGSCI commands are supported:

REGISTER EXTRACT extract database SCN #### SHARE AUTOMATIC
REGISTER EXTRACT extract database SCN #### SHARE extract
REGISTER EXTRACT extract database SHARE NONE
REGISTER EXTRACT extract database SCN #### SHARE NONE

Or

REGISTER EXTRACT extract DATABASE SHARE NONE
REGISTER EXTRACT extract DATABASE SCN #### SHARE NONE

In contrast, the following GGSCI commands are not supported in a downstream
configuration:

REGISTER EXTRACT extract DATABASE SHARE AUTOMATIC
REGISTER EXTRACT extract DATABASE SHARE extract


    AUTOMATIC
    Clone from the existing closest capture. If no suitable clone candidate
    is found, then a new build is created.

    extract
    Clone from the capture session associated for the specified extract. If
    this is not possible, then an error occurs the register does not complete.

    NONE
    Does not clone or create a new build; this is the default.

In a downstream configuration, the SHARE clause must be used in conjunction
with the SCN clause when registering for Extract.

Examples

Example 1

REGISTER EXTRACT sales LOGRETENTION

Example 2

REGISTER EXTRACT sales DATABASE

Example 3
```

```
REGISTER EXTRACT sales DATABASE CONTAINER (sales, finance, hr)

Example 4

REGISTER EXTRACT sales DATABASE ADD CONTAINER (customers)

Example 5

REGISTER EXTRACT sales DATABASE DROP CONTAINER (finance)

Example 6

REGISTER EXTRACT sales DATABASE SCN 136589

The beginning SCN of the dictionary build is 136589.

Example 7

REGISTER EXTRACT sales DATABASE SCN 67000 SHARE ext2

The valid start SCN, 67000 in this case; it is not necessarily the
current SCN.

Example 8

REGISTER EXTRACT sales DATABASE CONTAINER (sales, finance, hr)
SCN 136589

------------------------------------------------------------------------
SEND EXTRACT

Use SEND EXTRACT to communicate with a running Extract process. The
request is processed as soon as Extract is ready to accept commands from users.

Syntax

SEND EXTRACT group_name, {
BR {BRINTERVAL interval |
   BRSTART |
   BRSTOP | BRCHECKPOINT {IMMEDIATE | IN n{M|H} |
   AT yyyy-mm-dd hh:mm[:ss]]}} |
BR BRFSOPTION { MS_SYNC | MS_ASYNC }
CACHEMGR {CACHESTATS | CACHEQUEUES | CACHEVMUSAGE} |
FORCESTOP |
FORCETRANS transaction_ID [THREAD n] [FORCE] |

GETLAG |
GETPARAMINFO [parameter_name] [FILE output_file] |
GETTCPSTATS |
LOGEND |
LOGSTATS |
REPORT |
RESUME |
ROLLOVER |
SHOWTRANS [transaction_ID] [THREAD n] [COUNT n]
   [DURATION duration unit] [TABULAR]
   [FILE file_name [DETAIL]] |
SKIPTRANS transaction_ID [THREAD n] [FORCE] |
STATUS |
STOP |
TRACE[2] file_name |
```

```
TRACE[2] OFF |
TRACE OFF file_name |
TRACEINIT |
TRANSLOGOPTIONS INTEGRATEDPARAMS(parameter_specification)|
TRANLOGOPTIONS {PREPAREFORUPGRADETOIE | NOPREPAREFORUPGRADETOIE}
|TRANLOGOPTIONS
{PURGEORPHANEDTRANSACTIONS | NOPURGEORPHANEDTRANSACTIONS} |
TRANLOGOPTIONS TRANSCLEANUPFREQUENCY minutes |
VAMMESSAGE 'Teradata_command' |
VAMMESSAGE {'ARSTATS' | 'INCLUDELIST [filter]' | 'FILELIST [filter]'
| 'EXCLUDELIST
[filter]'} |
VAMMESSAGE 'OPENTRANS'
}

group_name

The name of the Extract group or a wildcard (*) to specify multiple
groups. For example, T* sends the command to all Extract processes whose
group names start with T. If an Extract is not running, an error is returned.
BR {BRINTERVALinterval| BRSTART | BRSTOP |
BRCHECKPOINT {IMMEDIATE | INn{H|M} | ATyyyy-mm-dd[ hh:mm[:ss]]}}

Sends commands that affect the Bounded Recovery mode of Extract.

    BRINTERVAL interval

    Sets the time between Bounded Recovery checkpoints. Valid values
    are from 20 minutes to 96 hours specified as M for minutes or H
    for hours, for example, 20M or 2H. The default interval is 4 hours.

    BRSTART

    Starts Bounded Recovery. This command should only be used under
    direction of Oracle Support.

    BRSTOP

    Stops Bounded Recovery for the run and for recovery. Consult Oracle
    Support before using this option. In most circumstances, when there
    is a problem with Bounded Recovery, it turns itself off.

    BRCHECKPOINT {IMMEDIATE | IN n{H|M} | AT yyyy-mm-dd[ hh:mm[:ss]]}}

    Sets the point at which a bounded recovery checkpoint is made.
    IMMEDIATE issues the checkpoint immediately when SEND EXTRACT is
    issued. IN issues the checkpoint in the specified number of hours
    or minutes from when SEND extract is issued. AT issues the checkpoint
    at exactly the specified time.

    BR BRFSOPTION {MS_SYNC | MS_ASYNC}

    Performs synchronous/asynchronous writes of the mapped data in
    Bounded Recovery.

        MS_SYNC
        Bounded Recovery writes of mapped data are synchronized for
        I/O data integrity completion.

        MS_ASYNC
        Bounded Recovery writes of mapped data are initiated or
```

```
    queued for servicing.

CACHEMGR {CACHESTATS | CACHEQUEUES | CACHEVMUSAGE}

Returns statistics about the Oracle GoldenGate memory cache
manager.

    CACHESTATS
    Returns all CACHEMGR statistics.

    CACHEQUEUES
    Returns statistics for the free queues only.

    CACHEVMUSAGE
    Returns statistics for the virtual memory use.

FORCESTOP

Forces Extract to stop, bypassing any notifications. This command
will stop the process immediately.

FORCETRANS transaction_ID [THREAD n] [FORCE]

Valid for MySQL and Oracle.

Forces Extract to write a transaction specified by its transaction ID
number to the trail as a committed transaction. FORCETRANS does not
commit the transaction to the source database. It only forces the
existing data to the trail so that it is processed (with an implicit
commit) by Replicat. You can repeat FORCETRANS for other transactions
in order of their age. Note that forcing a transaction to commit to
the trail (and therefore the target database) may cause data discrepancies
if the transaction is rolled back by the source user applications.

After using FORCETRANS, wait at least five minutes if you intend to issue
SEND EXTRACT with FORCESTOP. Otherwise, the transaction will still be
present.

If FORCETRANS is used immediately after Extract starts, you might
receive an error message that asks you to wait and then try the
command again. This means that no other transactions have been processed
yet by Extract. Once another transaction is processed, you will be
able to force the transaction to trail.

transaction_ID

The ID of the transaction. Get the transaction ID number with
SHOWTRANS or from an Extract runtime message. Extract ignores any
data added to the transaction after this command is issued. A
confirmation prompt must be answered unless FORCE is used. To use
FORCETRANS, the specified transaction must be the oldest one in the
list of transactions shown with SHOWTRANS.

THREAD n

Valid only for Oracle.

Use THREAD n to specify which thread generated the transaction in an Oracle
RAC environment if there are duplicate transaction IDs across threads.

FORCE
```

```
    Valid for Oracle. Use FORCE to bypass the confirmation prompt.

GETLAG

Determines a true lag time between Extract and the data source. Returns the
same results as LAG EXTRACT (see “LAG EXTRACT”).

GETPARAMINFO [parameter_name] [FILE output_file]

Use GETPARAMINFO to query runtime parameter values of a running instance,
including Extract, Replicat, and Manager. You can query for a single parameter
or all parameters and send the output to the console or a text file

    parameter_name

    The default behavior is to display all parameters in use, meaning those
    parameters that have ever been queried by the application, parameters,
    and their current values. If you specify a particular parameter, then the
    output is filtered by that name.

    FILEoutput_file

    The name of the text file that your output is redirected to.

GETTCPSTATS

Displays statistics about network activity between Extract and the target
system. The statistics include:

* Local and remote IP addresses.

* Inbound and outbound messages, in bytes and bytes per second.

* Number of receives (inbound) and sends (outbound). There will be at least
two receives per inbound message: one for the length and one or more for the
data.

* Average bytes per send and receive.

* Send and receive wait time: Send wait time is how long it takes for the write
to TCP to complete. The lower the send wait time, the better the performance
over
the network. Receive wait time is how long it takes for a read to complete.
Together, the send and receive wait times provide a rough estimate of network
round trip time. These are expressed in microseconds.

* Status of data compression (enabled or not).

* Uncompressed bytes and compressed bytes: When compared (uncompressed to
compressed),
these comprise the compression ratio, meaning how many bytes there were before
and
after compression. You can compare the compression ratio with the bytes that are
being compressed per second to determine if the compression rate is worth the
cost
in terms of resource and network consumption.

The TCPBUFSIZE option of RMTHOST and RMTHOSTOPTIONS controls the size of the TCP
buffer
for uncompressed data. What actually enters the network will be less than this
```

size
if compression is enabled. GETTCPSTATS shows post-compression throughput.

LOGEND

Confirms whether or not Extract has processed all of the records in the data source.
However, the following command is not supported on Oracle GoldenGate MySQL remote capture.

send extract_group logend

LOGSTATS

Valid only for Oracle.

Instructs Extract to issue a report about the statistics that are related to the processing of data from the Oracle redo log files. Extract uses an asynchronous log reader that reads ahead of the current record that Extract is processing, so that the data is available without additional I/O on the log files. The processing
is done through a series of read/write queues. Data is parsed by a producer thread
at the same time that additional data is being read from the log file by a reader
thread. Thus, the reason for the term "read-ahead" in the statistics.

The statistics are:

* AsyncReader.Buffersn: There is a field like this for each buffer queue that contains captured redo data. It shows the size, the number of records in it, and how long the wait time is before the data is processed. These statistics are given for write operations and read operations on the queue.

* REDO read ahead buffers: The number of buffers that are being used to read ahead asynchronously.

* REDO read ahead buffer size: The size of each buffer.

* REDO bytes read ahead for current redo: Whether read-ahead mode is on or off for the current redo log file (value of ON or OFF).

* REDO bytes read: The number of bytes read from all redo log files that are associated with this instance of Extract.

* REDO bytes read ahead: The number of bytes that were processed by the read-ahead mechanism.

* REDO bytes unused: The number of read-ahead bytes that were subsequently dropped as the result of Extract position changes or stale reads.

* REDO bytes parsed: The number of bytes that were processed as valid log data.

* REDO bytes output: The number of bytes that were written to the trail file (not including internal Oracle GoldenGate overhead).

REPORT

Generates an interim statistical report to the Extract report file. The statistics that are displayed depend upon the configuration of the STATOPTIONS

```
parameter when used with the RESETREPORTSTATS| NORESETREPORTSTATS option.

RESUME
Resumes (makes active) a process that was suspended by an EVENTACTIONS SUSPEND
event. The process resumes normal processing from the point at which it was
suspended.

ROLLOVER

Causes Extract to increment to the next file in the trail when restarting. For
example, if the current file is ET000002, the current file will be ET000003
after the command executes. A trail can be incremented from 000001through
999999, and then the sequence numbering starts over at 000000.

SHOWTRANS [transaction_ID] [THREAD n] [COUNT n]
[DURATION duration unit] [TABULAR] | [FILE file_name [DETAIL]]

Valid for MySQL and Oracle.

Displays information about open transactions. SHOWTRANS shows any of the
following, depending on the database type:

* Process checkpoint (indicating the oldest log needed to continue
processing the transaction in case of an Extract restart).  See Administering
Oracle GoldenGate for more information about checkpoints.


* Transaction ID

* Extract group name

* Redo thread number

* Timestamp of the first operation that Oracle GoldenGate extracts from
a transaction (not the actual start time of the transaction)

* System change number (SCN)

* Redo log number and RBA

* Status (Pending COMMIT or Running). Pending COMMIT is displayed while
a transaction is being written after a FORCETRANS was issued.


Without options, SHOWTRANS displays all open transactions that will fit
into the available buffer.

See the examples for sample output of SHOWTRANS. To further control output,
see the following options.

    transaction_ID
    Limits the command output to a specific transaction.

    THREAD n
    Valid only for Oracle.

    Constrains the output to open transactions against a specific Oracle
    RAC thread. For n, use a RAC thread number that is recognized by Extract.

    COUNT n
    Constrains the output to the specified number of open transactions,
```

```
    starting with the oldest one. Valid values are 1 to 1000.

    DURATION duration unit
    Restricts the output to transactions that have been open longer than
    the specified time, where:

    duration is the length of time expressed as a whole number.

    unit is one of the following to express seconds, minutes, hours, or days:

    S|SEC|SECS|SECOND|SECONDS
    M|MIN|MINS|MINUTE|MINUTES
    H|HOUR|HOURS
    D|DAY|DAYS

    TABULAR
    Valid only for Oracle.
    Generates output in tabular format similar to the default table printout
    from SQL*Plus. The default is field-per-row.

    FILE file_name [DETAIL]
    Valid for Oracle. Not valid for MySQL.

    Forces Extract to write the transaction information to the specified file.
    There is no output to the console.

    For Oracle, you can write a hex and plain-character dump of the data by
    using FILE with DETAIL. This dumps the entire transaction from memory to
    the file. Viewing the data may help you decide whether to skip the
transaction
    or force it to the trail.

    Note:
    Basic detail information is automatically written to the report file at
intervals
    specified by the WARNLONGTRANS CHECKINTERVAL parameter.

SKIPTRANS transaction_ID [THREAD n] [FORCE]
Valid for MySQL and Oracle.

Forces Extract to skip the specified transaction, thereby removing any current
data
from memory and ignoring any subsequent data. A confirmation prompt must be
answered
unless FORCE is used. After using SKIPTRANS, wait at least five minutes if you
intend
to issue SEND EXTRACT with FORCESTOP. Otherwise, the transaction will still be
present.
Note that skipping a transaction may cause data loss in the target database.

    Note:
    To use SKIPTRANS, the specified transaction must be the oldest one in the
    list of transactions shown with SHOWTRANS.

    transaction_ID
    The transaction ID number. Get the ID number with SHOWTRANS or from an
Extract
    runtime message.

    THREAD n
    Valid only for Oracle.
```

```
    Use THREAD n to specify which thread generated the transaction in an Oracle
RAC
    environment if there are duplicate transaction IDs. SKIPTRANS specifies the
    checkpoint index number, not the actual thread number. To specify the
correct
    thread, issue the INFO EXTRACT group_name SHOWCH command, and then specify
    the READ checkpoint index number that corresponds to the thread number that
    you want to skip. See the examples for details. See Administering Oracle
    GoldenGate for more information about checkpoints.

    FORCE
    Valid for Oracle. Not valid for MySQL

    Use FORCE to bypass the prompt that confirms your intent to skip the
transaction.

STATUS

Returns a detailed status of the processing state, including current position
and
activity. Possible processing status messages on the Current status line are:

* Delaying– waiting for more data

* Suspended– waiting to be resumed

* Processing data– processing data

* Starting initial load– starting an initial load task

* Processing source tables– processing data for initial load task

* Reading from data source– reading from the data source, such as a source
table or transaction log

* Adding record to transaction list– adding a record to the file memory
transaction list

* At EOF (end of file)– no more records to process

In addition to the preceding statuses, the following status notations appear
during an Extract recovery after an abend event. You can follow the progress
as Extract continually changes its log read position over the course of the
recovery.

* In recovery[1]– Extract is recovering to its checkpoint in the transaction
log.

* In recovery[2]– Extract is recovering from its checkpoint to the end of the
trail.

* Recovery complete – The recovery is finished, and normal processing will
resume.


STOP

Stops Extract. If there are any long-running transactions (based on the
WARNLONGTRANS parameter), the following message will be displayed:
```

```
Sending STOP request to EXTRACT JC108XT...
There are open, long-running transactions. Before you stop Extract, make the
archives containing data for those transactions available for when Extract
restarts. To force Extract to stop, use the SEND EXTRACT group, FORCESTOP
command.
Oldest redo log file necessary to restart Extract is:
Redo Thread 1, Redo Log Sequence Number 150, SCN 31248005, RBA 2912272.

TRACE[2] {file_name | OFF}
Turns tracing on and off. Tracing captures information to the specified file to
reveal processing bottlenecks. Contact Oracle Support for assistance if the
trace reveals significant processing bottlenecks.

    TRACE
    Captures step-by-step processing information.

    TRACE2
    Identifies code segments rather than specific steps.

    file_name
    Specifies the name of the file to which the trace information is written.
    If a trace is already running when SEND EXTRACT is issued with TRACE, the
    existing trace file is closed and the trace is resumed to the new file
    specified with file_name.

    OFF
    Turns off tracing.

TRACE OFF file_name

Turns tracing off only for the specified trace file.

TRACEINIT

Resets tracing statistics back to 0 and then starts accumulating statistics
again. Use this option to track the current behavior of processing, as
opposed to historical.

INTEGRATEDPARAMS(parameter_specification)

(Oracle) Supports an Integrated Extract. Sends a parameter specification to
the database inbound server while Extract is running in integrated mode. Only
one parameter specification can be sent at a time with this command. Parameter
changes do not take affect until the Extract is restarted.

To preserve the continuity of processing, the parameter change is made at a
transaction boundary. For a list of supported inbound server parameters, see
the Oracle GoldenGate documentation for your database.

TRANLOGOPTIONS{PREPAREFORUPGRADETOIE| NOPREPAREFORUPGRADETOIE}

(Oracle) Valid when upgrading from Classic to Integrated Extract on Oracle RAC.

When upgrading on Oracle RAC from Classic to Integrated Extract, you must set
the PREPAREFORUPGRADETOIE option before stopping Classic Extract for the upgrade
then wait for the information message in the report file that indicates that the
parameter has taken effect before proceeding with the upgrade. For detailed
upgrade instructions, see Upgrading Oracle GoldenGate for Windows and UNIX.

    PREPAREFORUPGRADETOIE
```

```
    Set PREPAREFORUPGRADETOIE in the Extract parameter file, which requires a
    restart of Extract, or you can set it dynamically for a running extract
    from GGSCI using this command:

    SEND EXTRACT extract_name TRANLOGOPTIONS
    PREPAREFORUPGRADETOIE

    NOPREPAREFORUPGRADETOIE

    Dynamically turns off the PREPAREFORUPGRADETOIE option if necessary.
    The default is NOPREPAREFORUPGRADETOIE.

TRANLOGOPTIONS {PURGEORPHANEDTRANSACTIONS |
NOPURGEORPHANEDTRANSACTIONS}

Valid for Oracle RAC. Enables or disables purging of orphaned transactions
that occur when a node fails and Extract cannot capture the rollback.

TRANLOGOPTIONS TRANSCLEANUPFREQUENCY minutes

Valid for Oracle RAC. Specifies the interval, in minutes, after which Oracle
GoldenGate scans for orphaned transactions and then re-scans to confirm and
delete them. Valid values are from 1 to 43200 minutes. Default is 10 minutes.

VAMMESSAGE 'Teradata_command'
VAMMESSAGE { 'ARSTATS' | 'INCLUDELIST [filter]' | 'EXCLUDELIST
[filter]' }
VAMMESSAGE 'OPENTRANS'
Sends a command to the capture API that is used by Extract.
A Teradata command can be any of the following:

    'control:terminate'
    Stops a replication group. Required before dropping or altering a
    replication group in Teradata.

    'control:suspend'
    Suspends a replication group. Can be used when upgrading Oracle GoldenGate.

    'control:resume'
    Resumes a replication group after it has been suspended.

    'control:copy database.table'
    Copies a table from the source database to the target database.



Examples

Example 1

SEND EXTRACT finance, ROLLOVER

Example 2

SEND EXTRACT finance, STOP

Example 3

SEND EXTRACT finance, VAMMESSAGE 'control:suspend'

Example 4
```

```
SEND EXTRACT finance, TRANLOGOPTIONS TRANSCLEANUPFREQUENCY 20
```

Example 5

This example explains SKIPTRANS. Start with the following SHOWCH output, which shows that thread 2 is at Read Checkpoint #3. See Administering Oracle GoldenGate
for more information about checkpoints.

```
INFO extract SHOWCH
Read Checkpoint #3
Oracle RAC Redo Log
Startup Checkpoint (starting position in the data source):
Thread #: 2
Sequence #: 17560
RBA: 65070096
Timestamp: 2017-07-30 20:04:47.000000
SCN: 1461.3499051750 (6278446271206)
Redo File: RAC4REDO/sss11g/onlinelog/group_4.292.716481937
```

Therefore, SKIPTRANS should be: SKIPTRANS xid THREAD 3.

Example 6

```
SEND EXTRACT finance, SHOWTRANS COUNT 2
```

Example 7

The following shows the default output of SHOWTRANS.

```
Oldest redo log file necessary to restart Extract is:
Redo Thread 1, Redo Log Sequence Number 148, SCN 30816254, RBA 17319664

XID        : 5.15.52582
Items        : 30000
Extract      : JC108XT
Redo Thread      : 1
Start Time      : 2017-01-18:12:51:27
SCN          : 20634955
Redo Seq      : 103
Redo RBA      : 18616848
Status      : Running

XID        : 7.14.48657
Items        : 30000
Extract        : JC108XT
Redo Thread    : 1
Start Time    : 2017-01-18:12:52:14
SCN        : 20635145
Redo Seq    : 103
Redo RBA    : 26499088
Status        : Running
```

Example 8

The following shows SHOWTRANS output with TABULAR in effect (view is truncated on right)

```
XID Items Extract Redo Thread Start Time 5.15.52582 30000 JC108XT 1
2017-01-18:12:52:14
```

```
Dumping transaction memory at 2017-01-21 13:36:54.
Record #1:
Header (140 bytes):
0: 0000 0A4A 0000 FFFF 0000 0000 0057 6C10 ...J.........Wl.
16: 02FF 3F50 FF38 7C40 0303 4141 414E 5A77 ..?P.8|@..AAANZw
32: 4141 4641 4141 4B6F 4941 4144 0041 4141 AAFAAAKoIAAD.AAA
48: 4E5A 7741 4146 4141 414B 6F49 4141 4400 NZwAAFAAAKoIAAD.
64: 4141 414E 5A77 414A 2F41 4142 7A31 7741 AAANZwAJ/AABz1wA
80: 4141 0041 4141 4141 4141 4141 4141 4141 AA.AAAAAAAAAAAAA
96: 4141 4141 4100 0000 0140 FF08 0003 0000 AAAAA....@......
112: 0000 0000 0000 70FF 0108 FFFF 0001 4A53 ......p.......JS
128: 554E 2E54 4355 5354 4D45 5200 UN.TCUSTMER.
Data (93 bytes):
0: 2C00 0400 0400 0000 0100 0200 0300 0000 ,...............
16: 0000 0000 0800 0000 1800 0000 2000 0400 ............ ...
32: 1000 0600 0200 0000 284A 414E 456C 6C6F ........(JANEllo
48: 6352 4F43 4B59 2046 4C59 4552 2049 4E43 cROCKY FLYER INC
64: 2E44 454E 5645 5220 6E43 4F20 7365 7400 .DENVER nCO set.
80: 0000 0000 0000 0C00 0000 0000 00 ..............
```

When analyzing the summary output of SHOWTRANS, understand that it shows all currently running transactions on the database (as many as will fit into a predefined buffer). Extract must track every open transaction, not just those that contain operations on tables configured for Oracle GoldenGate.

The Items field of the SHOWTRANS output shows the number of operations in the transaction that have been captured by Oracle GoldenGate so far, not the total number
of operations in the transaction. If none of the operations are for configured tables, or
if only some of them are, then Items could be 0 or any value less than the total number of operations.

The Start Time field shows the timestamp of the first operation that Oracle GoldenGate extracts from a transaction, not the actual start time of the transaction
itself

Note:
Command output may vary somewhat from the examples shown due ongoing enhancements
of Oracle GoldenGate.

START EXTRACT

Use START EXTRACT to start the Extract process. To confirm that Extract has started,
use the INFO EXTRACT or STATUS EXTRACT command. Extract can be started at its normal
start point (from initial or current checkpoints) or from an alternate, user-specified
position in the data source.

Normal Start Point

Without options, START EXTRACT directs a primary Extract and a data pump Extract to
start processing at one of the following locations in the data source to

maintain
data integrity:

* After graceful or abnormal termination: At the first unprocessed transaction in the data source from the previous run, as represented by the current read checkpoint.

* First-time startup after the group was created: At the start point specified with the ADD EXTRACT command.
See Administering Oracle GoldenGate for more information about checkpoints.

Extract also can be started from the command line of the operating system for certain synchronization configurations. For more information on the proper configuration and startup method to use for your purposes, see Administering Oracle GoldenGate.

Alternate Start Point

The ATCSN and AFTERCSN options enable you to establish a logical starting point for a primary Extract or a data pump, after you establish an approximate physical
starting point with the ADD EXTRACT or ALTER EXTRACT command. For example, in an initial-load scenario, after a backup is applied to the target, the serial identifier of the last transaction (such as an Oracle SCN) can be mapped to an Oracle GoldenGate CSN (commit sequence number) value, which can be used to start Extract with the AFTERCSN option. By starting with the first transaction after the specified CSN, Extract omits the transactions that were included in the backup, which would otherwise cause duplicate-record and missing-record errors.

Before starting Extract with ATCSN or AFTERCSN, you must establish a physical starting location with one of the following commands:

* ADD EXTRACT with the BEGIN option set to a timestamp that is earlier than the CSN value specified with ATCSN or AFTERCSN. The transaction log that contains the timestamp and every log thereafter must be available on the system before Extract is started.

* ALTER EXTRACT to the sequence number of the log that contains the CSN specified with ATCSN or AFTERCSN.

Note:
See Administering Oracle GoldenGate for more information about the values that comprise a CSN for a given database.

Syntax

START EXTRACT group_name [ATCSN csn | AFTERCSN csn]

group_name

The name of an Extract group or a wildcard (*) to specify multiple groups. For example, T* starts all Extract groups whose names begin with T.

ATCSN csn | AFTERCSN csn

Specifies an alternate start point. (See "Alternate Start Point" for usage instructions.)

ATCSN

```
    Directs Extract to position its start point at the first transaction that
    has the specified CSN. Any transactions in the data source that have CSN
    values less than the specified one are skipped.

    AFTERCSN
    Directs Extract to position its start point at the beginning of the first
    transaction after the one that has the specified CSN. Any transactions in
    the data source that have CSN values that are less than, or equal to, the
    specified one are skipped.

    csn
    Specifies a CSN value. Enter the CSN value in the format that is valid for
    the database. See Administering Oracle GoldenGate for CSN formats and
descriptions.
    Extract abends if the format is invalid and writes a message to the report
file.
    To determine the CSN to supply after an initial load is complete, use the
serial
    identifier at which the load utility completed. Otherwise, follow the
instructions
    in the initial load procedure for determining when to start Extract.

The following are additional guidelines to observe when using ATCSN and
AFTERCSN:

* To support starting at, or after, a CSN, the trail must be of Oracle
GoldenGate
version 10.0.0 or later, because the CSN is stored in the file header so that it
is available to downstream processes.

* When a record that is specified with a CSN is found, Extract issues a
checkpoint.
The checkpoint ensures that subsequent Extract startups begin from the requested
location, and not from a point prior to the requested CSN.

* You must establish a physical start point in the transaction log or trail for
Extract with ADD EXTRACT or ALTER EXTRACT before using ATCSN or AFTERCSN. These
options are intended to be an additional filter after Extract is positioned to
a physical location in the data source.


Examples

Example 1

START EXTRACT finance

Example 2

START EXTRACT finance ATCSN 684993

Example 3

START EXTRACT finance AFTERCSN 684993

------------------------------------------------------------------------
STATS EXTRACT

Use STATS EXTRACT to display statistics for one or more Extract groups. The
output
includes DML and DDL operations that are included in the Oracle GoldenGate
```

configuration.

To get the most accurate number of operations per second that are being processed,
do the following.

1. Issue the STATS EXTRACT command with the RESET option.

2. Issue the STATS EXTRACT REPORTRATE command. The LATEST STATISTICS field shows the operations per second.

Note:
The actual number of DML operations executed on a DB2 database might not match the number of extracted DML operations reported by Oracle GoldenGate. DB2 does not log update statements if they do not physically change a row, so Oracle GoldenGate cannot detect them or include them in statistics.

To get accurate statistics on a Teradata source system where Oracle GoldenGate is configured in maximum protection mode, issue STATS EXTRACT to the VAM-sort Extract, not the primary Extract. The primary Extract may contain statistics for uncommitted transactions that could be rolled back; whereas the VAM-sort Extract reports statistics only for committed transactions.

Syntax

```
STATS EXTRACT group_name
 statistic]
 TABLE [container. | catalog.]schema.table]
 TOTALSONLY [container. | catalog.]schema.table]
 REPORTCHARCONV]
 REPORTFETCH | NOREPORTFETCH]
 REPORTRATE time_units]
 ... ]
```

group_name

The name of an Extract group or a wildcard (*) to specify multiple groups. For example, T* returns statistics for all Extract groups whose names start with T.

statistic

The statistic to be displayed. More than one statistic can be specified by separating each with a comma, for example STATS EXTRACT finance, TOTAL, DAILY.

Valid values:

    TOTAL
    Displays totals since process startup.

    DAILY
    Displays totals since the start of the current day.

    HOURLY
    Displays totals since the start of the current hour.

    LATEST
    Displays totals since the last RESET command.

    RESET
    Resets the counters in the LATEST statistical field.

```
TABLE [container. | catalog.]schema.table

Displays statistics only for the specified table or a group of tables
specified with a wildcard (*). The table name or wildcard specification
must be fully qualified with the two-part or three-part name, for example
hr.empor *.*.*.

TOTALSONLY [container. | catalog.]schema.table

Summarizes the statistics for the specified table or a group of tables
specified with a wildcard (*). The table name or wildcard specification
must be fully qualified with the two-part or three-part name, for example
hr.empor *.*.*.

REPORTCHARCONV

Use only when TABLE parameters have a TARGET clause and character-set
conversion is performed. The following statistics are added to the STATS
output:

Total column character set conversion failure: the number of validation
or conversion failures in the current Extract run.

Total column data truncation: the number of times that column data was
truncated in the current Extract run as the result of character set conversion

REPORTFETCH | NOREPORTFETCH

Controls whether or not statistics about fetch operations are included
in the output. The default is NOREPORTFETCH. See “STATOPTIONS” for defaults
that control fetching and options for altering fetch behavior. The output of
REPORTFETCH is as follows:

* row fetch attempts: The number of times Extract attempted to fetch a column
value from the database when it could not obtain the value from the transaction
log.

* fetch failed: The number of row fetch attempts that failed.

* row fetch by key: Valid for Oracle. The number of row fetch attempts that
were made by using the primary key. The default is to fetch by row ID.


REPORTRATE time_units

Displays statistics in terms of processing rate rather than absolute values.

Valid values:

HR
MIN
SEC

Example

Example 1

The following example displays total and hourly statistics per minute for a
specific table, and it also resets the latest statistics and outputs fetch
statistics.
```

```
STATS EXTRACT finance, TOTAL, HOURLY, TABLE hr.acct, REPORTRATE MIN, RESET,
REPORTFETCH

Example 2

The following is sample output using the LATESTand REPORTFETCH options

STATS EXTRACT ext, LATEST, REPORTFETCH
Sending STATS request to EXTRACT GGSEXT...
Start of Statistics at 2017-01-08 11:45:05.
DDL replication statistics (for all trails):
*** Total statistics since extract started ***

    Operations         3.00
    Mapped operations     3.00
     Unmapped operations     0.00
     Default operations     0.00
     Excluded operations     0.00

Output to ./dirdat/aa:
Extracting from HR.EMPLOYEES to HR.EMPLOYEES:
*** Latest statistics since 2017-01-08 11:36:55 ***

     Total inserts         176.00
     Total updates         0.00
     Total deletes         40.00
     Total discards         0.00
     Total operations     216.00

Extracting from HR.DEPARTMENTS to HR.DEPARTMENTS:
*** Latest statistics since 2017-01-08 11:36:55 ***
No database operations have been performed.
End of Statistics.

------------------------------------------------------------------------
STATUS EXTRACT

Use STATUS EXTRACT to determine whether or not Extract is running. A status
of RUNNING can mean one of the following:

* Active: Running and processing (or able to process) data. This is the
normal state of a process after it is started.

* Suspended: The process is running, but suspended due to an EVENTACTIONS SUSPEND
action. In a suspended state, the process is not active, and no data can be
processed, but the state of the current run is preserved and can be continued by
issuing the RESUME command in GGSCI. The RBA in the INFO command reflects the
last
checkpointed position before the suspend action. To determine whether the state
is active or suspended, issue the SEND EXTRACT command with the STATUS option.


Syntax

STATUS EXTRACT group_name  TASKS | ALLPROCESSES] [UPGRADE | DOWNGRADE]

group_name

The name of an Extract group or a wildcard (*) to specify multiple groups.
For example, T* returns status for all Extract groups whose names begin with T.
```

```
TASKS

Displays status only for Extract tasks. By default, tasks are not displayed
unless you specify a single Extract group (without wildcards).

ALLPROCESSES

Displays status for all Extract groups, including tasks.

UPGRADE | DOWNGRADE

Valid for an Oracle Database only. If Extract cannot be upgraded or downgraded,
the reason why is displayed. A wildcarded Extract name is not allowed with this
option. Before using this command, issue the DBLOGIN command.

    UPGRADE
    Displays whether the Extract can be upgraded from classic capture mode to
    integrated capture mode.

    DOWNGRADE
    Displays whether the Extract can be downgraded from integrated capture
    mode to classic capture mode.

Examples

Example 1

STATUS EXTRACT finance

Example 2

STATUS EXTRACT fin*

------------------------------------------------------------------------
STOP EXTRACT

Use STOP EXTRACT to stop Extract gracefully. The command preserves the state
of synchronization for the next time Extract starts, and it ensures that
Manager does not automatically start Extract.

If there are open, long-running transactions when you issue STOP EXTRACT, you
might be advised of the oldest transaction log file that will be needed for
that transaction when Extract is restarted. You can use the SEND EXTRACT option
of SHOWTRANS to view details and data of those transactions and then, if
desired,
use the SKIPTRANS or FORCETRANS options to skip the transaction or force it to
be
written as a committed transaction to the trail. See “SEND EXTRACT”.

Syntax

STOP EXTRACT group_name

group_name

The name of an Extract group or a wildcard (*) to specify multiple groups.
For example, T* stops all Extract processes for groups whose names begin with T.

STOP EXTRACT finance
```

```
------------------------------------------------------------------
FORCEAPPEND

FORCEAPPEND instructs Extracts, primary or data pump, to forgive the missing CSN
information in existing output trails during recovery stage, so that the new
trail files can be seemingly appended to the existing ones.

Syntax

START data_pump FORCEAPPEND

data_pump

The name of the Extract data pump for which the FORCEAPPEND option needs to be
applied.

Example:

START EXT1PMP FORCEAPPEND

------------------------------------------------------------------
UNREGISTER EXTRACT

UNREGISTER EXTRACT

Use UNREGISTER EXTRACT to remove the registration of an Extract group from an
Oracle Database. UNREGISTER EXTRACT is valid only for a primary Extract group.
Do not use it for a data pump Extract.

To register an Extract group with the database, use the REGISTER EXTRACT
command.

To upgrade an Extract from classic capture mode to integrated capture mode, use
the ALTER EXTRACT command.

See the Oracle GoldenGate documentation for your database for more
information about configuring and registering Oracle GoldenGate for
Oracle Database.

Syntax

UNREGISTER EXTRACT group_name
{DATABASE | LOGRETENTION}

group_name

The name of the Extract group that is to be unregistered from the database. Do
not use a wildcard. This group must currently be registered with the database.

DATABASE

Disables integrated capture mode for the Extract group.

This command removes the database capture (mining) server that has the same name
as the Extract group. For additional information about support for, and
configuration of, the Extract capture modes, see the Oracle GoldenGate
documentation
for your database.

Before using UNREGISTER EXTRACT with DATABASE, do the following:
```

```
1. Stop Extract with the STOP EXTRACT command.

2. Log in to the mining database with the DBLOGIN or MININGDBLOGIN command with
the
privileges granted in the dbms_goldengate_auth.grant_admin_privilege procedure.
For local capture, DBLOGIN is required. For downstream capture, DBLOGIN and
MININGDBLOGIN are both required.

3. Delete the Extract group with DELETE EXTRACT.


LOGRETENTION

Disables log retention for the specified Extract group and removes the
underlying
Oracle Streams capture process. Use UNREGISTER EXTRACT with LOGRETENTION only if
you no longer want to capture changes with this Extract group. The log-retention
feature is controlled with the LOGRETENTION option of the TRANLOGOPTIONS
parameter.

Before using UNREGISTER EXTRACT with LOGRETENTION, stop Extract with the STOP
EXTRACT
command. Next, issue the DBLOGIN command with the privileges shown in Examples 1-
2.

Examples

Example 1

UNREGISTER EXTRACT sales LOGRETENTION

Example 2

UNREGISTER EXTRACT sales DATABASE

------------------------------------------------------------------------
####################################
#
#REPLICAT COMMANDS
#
#
####################################
------------------------------------------------------------------------
ADD REPLICAT

Use ADD REPLICAT to create a Replicat group. Unless SPECIALRUN is specified,
ADD REPLICAT creates an online process group that creates checkpoints so
that processing continuity is maintained from run to run.

This command cannot exceed 500 bytes in size for all keywords and input,
including any text that you enter for the DESC option.

Oracle GoldenGate supports up to 5,000 concurrent Extract and Replicat groups
per instance of Oracle GoldenGate Manager. At the supported level, all groups
can be controlled and viewed in full with GGSCI commands such as the INFO and
STATUS commands. Oracle GoldenGate recommends keeping the number of Extract and
Replicat groups (combined) at the default level of 300 or below in order to
manage your environment effectively.

(Oracle) Unless the INTEGRATED option is used, this command creates a Replicat
group in non-integrated mode.
```

Syntax

```
ADD REPLICAT group_name
 INTEGRATED | COORDINATED [MAXTHREADS number]]
{, SPECIALRUN |
   , EXTFILE file_name |
   , EXTTRAIL trail_name}
 BEGIN {NOW | yyyy-mm-dd[ hh:mm[:ss[.cccccc]]]} |
   , EXTSEQNO sequence_number, EXTRBA rba]
{PARALLEL | PARALLEL INTEGRATED EXTTRAIL trail_name ,
 CHECKPOINTTABLE owner.table | NODBCHECKPOINT}
 PARAMS file_name]
 REPORT file_name]
 DESC 'description']
```

group_name

The name of the Replicat group. The name of a coordinated Replicat group can contain a maximum of five characters. The name of a regular Replicat group can contain up to eight characters. See Administering Oracle GoldenGate for more information about naming conventions for process groups.

INTEGRATED

(Oracle) Creates the Replicat in integrated mode. Without this option, ADD REPLICAT creates the Replicat in non-integrated (classic) mode. In this mode, the Replicat process leverages the apply processing functionality that is available within the Oracle Database. In this mode, Replicat operates as follows:

* Reads the Oracle GoldenGate trail.

* Performs data filtering, mapping, and conversion.

* Constructs logical change records (LCR) that represent source database DML or DDL transactions (in committed order).

* Attaches to a background process in the target database known as a database inbound server by means of a lightweight streaming interface.

* Transmits the LCRs to the inbound server, which applies the data to the target database.

Do not use INTEGRATED with the SPECIALRUN or EXTFILE options. INTEGRATED must be used for an online change-synchronization Replicat that reads from a local EXTTRAIL-specified trail.
Integrated Replicat does not require a checkpoint table (ADD CHECKPOINTTABLE command and CHECKPOINTTABLE parameter) or a trace table (TRACETABLE parameter). Integrated Replicat does not maintain either of these tables.
When in integrated mode, Replicat does not support the following parameters:

* BULKLOAD(Do not use integrated Replicat as an initial-load Replicat.)

* SPECIALRUN

* GENLOADFILES

* SHOWSYNTAX

* MAXTRANSOPS(is ignored)

See the Oracle GoldenGate documentation for your database for more information about configuring and using integrated Replicat.

COORDINATED [MAXTHREADS number]

Creates the Replicat in coordinated mode. A coordinated Replicat is multithreaded to enable parallel processing. This option adds the coordinator (identified by the group name itself) and the maximum number of processing threads that are specified by default or with MAXTHREADS. Dependencies are computed and coordinated by the coordinator, and the SQL processing is performed by the threads.

Do not use COORDINATED with the SPECIALRUN or EXTFILE options. COORDINATED must be used for an online change-synchronization Replicat that reads from a local EXTTRAIL-specified trail. For more information about coordinated Replicat, see Administering Oracle GoldenGate.

Note:
Note that the group name of a coordinated Replicat can contain only five characters. See Administering Oracle GoldenGate for more information about naming conventions for process groups.

MAXTHREADS number
Specifies the maximum number of processing threads that this Replicat group can spawn. These threads are all created on startup, but depending on what is specified in the MAP statements in the parameter file, some or all of these threads will process the workload at any given time. As a general rule, specify twice the number of threads that you specify in the MAP statements when you partition the workload. This allows you to add threads in the event that the workload increases, without having to drop and recreate the Replicat group. See Administering Oracle GoldenGate for for more information about how to partition the workload across threads.

The default number of threads is 25 if MAXTHREADS is omitted. The maximum number of threads is 500.

MAXTHREADS has a relationship to the MAXGROUPS parameter. MAXGROUPS controls the maximum number of process groups (Extract and Replicat) allowed per instance of Oracle GoldenGate. Each Replicat thread is considered a Replicat group in the context of MAXGROUPS. Therefore, the number of Extract and Replicat groups in the Oracle GoldenGate instance, plus the value of MAXTHREADS, cannot exceed the value of MAXGROUPS.

SPECIALRUN
Creates a Replicat special run as a task. Either SPECIALRUN, EXTFILE, or EXTTRAIL is required. When Extract is in SPECIALRUN mode, do not start Replicat with the START REPLICAT command in GGSCI. Do not use this option with the INTEGRATED or COORDINATED option.

EXTFILE file_name
Specifies the relative or fully qualified name of an Extract file that is specified with RMTFILE in the Extract parameter file. Do not use this option with the INTEGRATED option.

EXTTRAIL trail_name
Specifies the relative or fully qualified name of a trail that was created

with the ADD RMTTRAIL or ADD EXTTRAIL command.

BEGIN {NOW | yyyy-mm-dd[hh:mm[:ss[.ccccccc]]]}
Defines an initial checkpoint in the trail.

    NOW
    Begins replicating changes from the time when the group is created.

    yyyy-mm-dd[hh:mm[:ss[.ccccccc]]]
    Begins extracting changes from a specific time.

EXTSEQNO sequence_number

Specifies the sequence number of the file in a trail in which to begin processing data. Specify the sequence number, but not any zeroes used for padding. For example, if the trail file is c:\ggs\dirdat/aa000000026, you would specify EXTSEQNO 26.
By default, processing begins at the beginning of a trail unless this option is used. To use EXTSEQNO, you must also use EXTRBA. Contact Oracle Support before using this option.

EXTRBA rba

Specifies the relative byte address within the trail file that is specified by EXTSEQNO. Contact Oracle Support before using this option.

PARALLEL

Adds the Replicat in parallel mode. In this mode, Replicat applies transactions in
parallel to improve the performance. It takes into account dependencies between transactions. PARALLEL INTEGRATED adds the parallel Replicat in integrated mode, which like Integrated Replicat leverages the apply processing functionality that is
available within the Oracle Database. You must use a checkpointtable with this Replicat.

CHECKPOINTTABLE owner.table

Not valid for Oracle GoldenGate Applications Adapter or Oracle GoldenGate Big Data.

Specifies that this Replicat group will write checkpoints to the specified table in the database. Include the owner and table name, as in hr.hr_checkpoint. This argument overrides any default CHECKPOINTTABLE specification in the GLOBALS file. The table must first be added with the ADD CHECKPOINTTABLE command. Do not use this option with the INTEGRATED option. When NODBCHECKPOINT is specified, an additional checkpoint file for Java is not created.

NODBCHECKPOINT

Specifies that this Replicat group will not write checkpoints to a checkpoint table. This argument overrides any default CHECKPOINTTABLE specification in the GLOBALS file. This argument is required if you do not want to use a checkpoint table with the Replicat group that is being created. Do not use this option with the INTEGRATED option.

PARAMS file_name

Specifies a parameter file in a location other than the default of dirprm within the Oracle GoldenGate directory. Specify the fully qualified path name.

```
REPORT file_name

Specifies the full path name of a process report file in a location other
than the default of dirrpt within the Oracle GoldenGate directory.

DESC 'description'

Specifies a description of the group, such as 'Loads account_tab on Serv2'.
Enclose the description within quotes. You can use either the abbreviated
keyword DESC or the full word DESCRIPTION.

Examples

Example 1

ADD REPLICAT sales, EXTTRAIL dirdat/rt

Example 2

ADD REPLICAT sales, INTEGRATED, EXTTRAIL dirdat/rt

Example 3

This example creates Replicat in coordinated mode. It indicates that up
to 100 threads can be employed in parallel at any given point in processing.

ADD REPLICAT sales, COORDINATED MAXTHREADS 100, EXTTRAIL dirdat/rt

Example 4

ADD REPLICAT sales, PARALLEL, EXTTRAIL dirdat/rt , checkpointtable
  ggadmin.checkpoint
Example 5

This example creates parallel Replicat in integrated mode.

ADD REPLICAT sales, PARALLEL INTEGRATED, EXTTRAIL dirdat/rt ,
  checkpointtable ggadmin.checkpoint

------------------------------------------------------------------------
ALTER REPLICAT

Use ALTER REPLICAT to change the attributes of a Replicat group that was
created with the ADD REPLICAT command. Before using this command, stop
Replicat by issuing the STOP REPLICAT command. If this is a coordinated
Replicat group, the ALTER takes effect for all threads unless the threadID
option is used.

    Note:
    ALTER REPLICAT does not support switching from regular Replicat mode to
    coordinated mode. You must stop processes, make certain all of the en route
    data is applied to the target, roll the trail to a new trail, drop and
    recreate the Replicat group in coordinated mode, and then start the
processes
    again.

Syntax

ALTER REPLICAT group_name[threadID], {
ADD REPLICAT option  ...] |
```

```
 INTEGRATED | NONINTEGRATED, CHECKPOINTTABLE owner.table }

group_name[threadID]

The name of the Replicat group or a thread of a coordinated Replicat that is
to be altered. To specify a thread, use the full thread name, such as ALTER
REPLICAT fin003, EXTSEQNO 53. If a thread ID is not specified, the ALTER takes
effect for all threads of the Replicat group.

ADD REPLICAT option

An ADD REPLICAT option. For a non-integrated Replicat, you can change the
description or any service option that was configured with the ADD REPLICAT
command, except for the CHECKPOINT and NODBCHECKPOINT options.

INTEGRATED

Switches Replicat from non-integrated mode to integrated mode. Transactions
currently in process are applied before the switch is made. See Administering
Oracle GoldenGate for the full procedure for performing the transition from
non-integrated to integrated Replicat.

NONINTEGRATED, CHECKPOINTTABLE owner.table

(Oracle) Switches Replicat from integrated mode to non-integrated mode.

For CHECKPOINTTABLE, specify the owner and name of a checkpoint table. This
table must be created with the ADD CHECKPOINTTABLE command before issuing ALTER
EXTRACT with NONINTEGRATED.

See Administering Oracle GoldenGate for the full procedure for performing
the transition from integrated Replicat to non-integrated Replicat.

See the Oracle GoldenGate documentation for your database for more information
about integrated Replicat.

Examples

Example 1

ALTER REPLICAT finance, EXTSEQNO 53

Example 2

ALTER REPLICAT finance, EXTRBA 0

Example 3

ALTER REPLICAT finance, BEGIN 2017-01-07 08:00:00

Example 4

ALTER REPLICAT finance, INTEGRATED

Example 5

ALTER REPLICAT finance, NONINTEGRATED, CHECKPOINTTABLE ogg.checkpt

Example 6

ALTER REPLICAT fin001, EXTSEQNO 53
```

```
------------------------------------------------------------------
CLEANUP REPLICAT

Use CLEANUP REPLICAT to delete run history for a specified Replicat group.
The cleanup keeps the last run record intact so that Replicat can resume
processing from where it left off.

Before using this command, stop Replicat by issuing the STOP REPLICAT command.

Syntax

CLEANUP REPLICAT group_name[threadID]  SAVE count]

group_name[threadID]

One of the following:

* group_name: The name of a Replicat group or a wildcard (*) to specify
multiple groups. For example, T* cleans up all Replicat groups whose names
begin with T. If the specified group (or groups) is a coordinated Replicat,
the cleanup applies to all threads.

* group_namethreadID: A thread of a coordinated Replicat, identified by its
full name (group name plus threadID), such as finance003.


SAVE count

Excludes the specified number of the most recent records from the cleanup.

Examples

Example 1

The following deletes all but the last record.

CLEANUP REPLICAT finance

Example 2

The following deletes all but the most recent five records.

CLEANUP REPLICAT *, SAVE 5

Example 3

The following deletes all but the most recent five records for thread 3
of coordinated Replicat group fin.

CLEANUP REPLICAT fin003, SAVE 5

------------------------------------------------------------------
DELETE REPLICAT

Use DELETE REPLICAT to delete a Replicat group. This command deletes the
checkpoint file but leaves the parameter file intact. Then you can re-create
the group or delete the parameter file as needed. This command frees up
trail files for purging by Manager, because the checkpoints used by the
deleted group are removed (assuming no other processes are reading the file).
```

Before using DELETE REPLICAT, stop Replicat with the STOP REPLICAT command.

If this is an integrated Replicat (Oracle only) or a non-integrated Replicat that uses a checkpoint table, do the following after you stop Replicat:

1. Log into the database by using the DBLOGIN command. DBLOGIN enables DELETE REPLICAT to delete the checkpoints from the checkpoint table of a non-integrated Replicat or to delete the inbound server that an integrated Replicat uses. For more information, see “DBLOGIN”.

2. Issue DELETE REPLICAT.

Syntax

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

group_name

The name of a Replicat group or a wildcard (*) to specify multiple groups. For example, T* deletes all Replicat groups whose names begin with T.

!
Use this option to force the Replicat group to be deleted if the DBLOGIN command is not issued before the DELETE REPLICAT command is issued. If the group is a nonintegrated Replicat, this option deletes the group's checkpoints from the checkpoint file on disk, but not from the checkpoint table in the database. If using this option to delete an integrated Replicat group, you must use the UNREGISTER REPLICAT command to delete the inbound server from the target database. This option can also be used to ignore the prompt that occurs when a wildcard specifies multiple groups.

Note:
The basic DELETE REPLICAT command commits an existing Replicat transaction, but the ! option prevents the commit.

Example

```
DELETE REPLICAT finance
```

INFO REPLICAT

Use INFO REPLICAT to retrieve the processing history of a Replicat group. The output of this command includes:

* The status of Replicat (STARTING, RUNNING, STOPPED or ABENDED). STARTING means that the process has started but has not yet locked the checkpoint file for processing.

* (Oracle Database) The Replicat mode: non-integrated or integrated.

* Whether or not Replicat is in coordinated mode and, if so, how many threads it currently uses.

* Approximate Replicat lag.

* The trail from which Replicat is reading.

* Replicat run history, including checkpoints in the trail.

* Information about the Replicat environment.

The basic command displays information only for online (continuous) Replicat groups. Tasks are excluded.
Replicat can be stopped or running when INFO REPLICAT is issued. In the case of a running process, the status of RUNNING can mean one of the following:

* Active: Running and processing (or able to process) data. This is the normal state of a process after it is started.

* Suspended: The process is running, but suspended due to an EVENTACTIONS SUSPEND action. In a suspended state, the process is not active, and no data can be processed, but the state of the current run is preserved and can be continued by issuing the RESUME command in GGSCI. The RBA in the INFO command reflects the last checkpointed position before the suspend action. To determine whether the state is active or suspended, issue the SEND REPLICAT command with the STATUS option.

About Lag

Checkpoint Lag is the lag, in seconds, at the time the last checkpoint was written to the trail. For example, consider the following example.

* Current time = 15:00:00

* Last checkpoint = 14:59:00

* Timestamp of the last record processed =14:58:00

Assuming these values, the lag is reported as 00:01:00 (one minute, the difference between 14:58 and 14:59).

A lag value of UNKNOWN indicates that Replicat could be running but has not yet processed records, or that the source system's clock is ahead of the target system's clock (due to clock imperfections, not time zone differences). For more precise lag information, use LAG REPLICAT.

Syntax

```
INFO REPLICAT group_name[thread id]
 [DETAIL]
 [SHOWCH [n]]
 [TASKS | ALLPROCESSES]
```

group_name[thread id]

The name of:

* A Replicat group or a wildcard (*) to specify multiple groups. For example, T* shows information for all Replicat groups whose names begin with T.

* A thread of a coordinated Replicat, identified by its full name. For example, fin003 shows information only for thread 3 of the fin group.

DETAIL

Displays detail information. For an Oracle target, DETAIL displays the name of the inbound server when Replicat is in integrated mode. See Example 4.

To view LOGBSN information with the DETAIL output, issue the DBLOGIN command before you issue INFO REPLICAT. If the command is issued for a specific thread ID of a coordinated Replicat, only the LOGBSN for that thread is displayed. Otherwise, the LOGBSNs for all threads are displayed. For more information about recovering Extract by using the LOGBSN, see Administering Oracle GoldenGate.

If Replicat is in coordinated mode, DETAIL will display only the active threads. For example, if a Replicat named CR was created with a maximum of 15 threads, but only threads 7-9 are running, INFO REPLICAT group_name with DETAIL will show only the coordinator thread (CR), CR007, CR008, and CR009. Checkpoints will exist for the other threads, but they will not be shown in the command output. See the examples for sample output.

SHOWCH [n]

Displays current checkpoint details, including those recorded to the checkpoint file and those recorded to the checkpoint table, if one is being used. The database checkpoint display includes the table name, the hash key (unique identifier), and the create timestamp.

Specify a value for n to include the specified number of previous checkpoints as well as the current one.

TASKS

Displays only Replicat tasks. Tasks that were specified by a wildcard argument are not displayed by INFO REPLICAT.

ALLPROCESSES

Displays all Replicat groups, including tasks.

Examples

Example 1

```
INFO REPLICAT *, DETAIL, ALLPROCESSES
```

Example 2

```
INFO REPLICAT *, TASKS
```

Example 3

```
INFO REPLICAT fin003, SHOWCH
```

Example 4

The following shows sample output of INFO REPLICAT with DETAIL.

```
REPLICAT DELTPCC         Last Started 2017-01-21 11:40 Status RUNNING
Checkpoint Lag             00:00:00 (updated 232:39:41 ago)
Log Read Checkpoint File     C:\GGS\DIRDAT\RT000000000
                 2017-01-21 18:54:33.000000 RBA 4735245

Extract Source          Begin             End
```

```
C:\GGS\DIRDAT\RT000000000  2017-01-21 18:54      2017-01-21 18:54
C:\GGS\DIRDAT\RT000000000  * Initialized *      2017-01-21 18:54

Current directory      C:\GGS
Report file          C:\GGS\dirrpt\DELTPCC.rpt
Parameter file      dirprm\DELTPCC.prm
Checkpoint file      C:\GGS\dirchk\DELTPCC.cpr
Checkpoint table      GG.CHECKPT
Process file          C:\GGS\dirpcs\DELTPCC.pcr
Error log          C:\GGS\ggserr.log

Example 5

The following shows INFO EXTRACT with DETAIL for a coordinated Replicat.

GGSCI (sysa) 3 info ra detail

REPLICAT RA         Last Started 2017-05-01 14:15 Status RUNNING
COORDINATED         Coordinator MAXTHREADS 15
Checkpoint Lag         00:00:00 (updated 00:00:07 ago)
Process ID         11445
Log Read Checkpoint     File ./dirdat/withMaxTransOp/bg000000001
             2017-05-02 07:49:45.975662 RBA 44704

Lowest Log BSN value: (requires database login)

Active Threads:
ID Group Name PID Status Lag at Chkpt Time Since Chkpt
1  RA001  11454 RUNNING  00:00:00  00:00:01
2  RA002  11455 RUNNING  00:00:00  00:00:04
3  RA003  11456 RUNNING  00:00:00  00:00:01
5  RA005  11457 RUNNING  00:00:00  00:00:02
6  RA006  11458 RUNNING  00:00:00  00:00:04
7  RA007  11459 RUNNING  00:00:00  00:00:04

Current directory /scratch/vara/view_storage/vara_gg7/work/worklv/oggora1
Report file /scratch/vara/view_storage/vara_gg7/work/worklv/oggora1/dirrpt/
RA.rpt
Parameter file /net/slc03jgo/scratch/vara/view_storage/vara_gg7/work/worklv/
oggora1/dirprm/ra.prm
Checkpoint file /scratch/vara/view_storage/vara_gg7/work/worklv/oggora1/dirchk/
RA.cpr Checkpoint table atstgt.checkPoint_ra
Process file /scratch/vara/view_storage/vara_gg7/work/worklv/oggora1/dirpcs/
RA.pcr
Error log /scratch/vara/view_storage/vara_gg7/work/worklv/oggora1/
ggserr.log

Example 6

The following shows INFO EXTRACT with DETAIL for a threadID of a coordinated
Replicat.

GGSCI (sysa) 5 info ra002 detail

REPLICAT RA002         Last Started 2017-05-01 14:15     Status RUNNING
COORDINATED         Replicat Thread         Thread 2
Checkpoint Lag 00:00:00 (updated 00:00:02 ago)
Process         ID 11455
Log Read Checkpoint     File ./dirdat/withMaxTransOp/bg000000001
            2017-05-01 14:13:37.000000 RBA 44704
```

```
Current Log BSN value: (requires database login)

 Extract Source             Begin             End
./dirdat/withMaxTransOp/bg000000001     2017-05-01 14:11     2017-05-01 14:13
./dirdat/withMaxTransOpbg000000001     2017-05-01 14:11     2017-05-01 14:11
./dirdat/withMaxTransOp/bg000000001     * Initialized *     2017-05-01 14:11
./dirdat/withMaxTransOp/bg000000001     * Initialized *     First Record
Current directory     /scratch/vara/view_storage/vara_gg7/work/worklv/oggora1
Report file         /scratch/vara/view_storage/vara_gg7/work/worklv/oggora1/
dirrpt/ RA002.rpt
Parameter file         /net/slc03jgo/scratch/vara/view_storage/vara_gg7/work/
worklv/ oggora1/dirprm/ra.prm
Checkpoint file      /scratch/vara/view_storage/vara_gg7/work/worklv/oggora1/
dirchk/
RA002.cpr
Checkpoint table      atstgt.checkPoint_ra
Process file          /scratch/vara/view_storage/vara_gg7/work/worklv/oggora1/
dirpcs/
RA002.pcr
Error log          /scratch/vara/view_storage/vara_gg7/work/worklv/oggora1/
ggserr.log

------------------------------------------------------------------------
KILL REPLICAT

Use KILL REPLICAT to kill a Replicat process. Killing a process leaves the
most recent checkpoint in place, and the current transaction is rolled back
by the database, guaranteeing that no data is lost when the process is
restarted. The Manager process will not attempt to restart a killed Replicat
process. Use this command only if Replicat cannot be stopped gracefully with
the STOP REPLICAT command.

Syntax

KILL REPLICAT group_name

group_name

The name of a Replicat group or a wildcard (*) to specify multiple groups.
For example, T* kills all Replicat processes whose group names begin with T.

Example

KILL REPLICAT finance

------------------------------------------------------------------------
LAG REPLICAT

Use LAG REPLICAT to determine a true lag time between Replicat and the trail.
LAG REPLICAT estimates the lag time more precisely than INFO REPLICAT because
it communicates with Replicat directly rather than reading a checkpoint
position.

For Replicat, lag is the difference, in seconds, between the time that the last
record was processed by Replicat (based on the system clock) and the timestamp
of the record in the trail.

If the heartbeat functionality is enable, you can view the associated lags.
A DBLOGIN is required to view the heartbeat lag.

Syntax
```

```
LAG REPLICAT

 [group_name[threadID]name]
 [GLOBAL]

group_name[threadID]
The name of:

* A Replicat group or a wildcard (*) to specify multiple groups. For
example, T* shows lag for all Replicat groups whose names begin with T.

* A thread of a coordinated Replicat, identified by its full name. For
example, fin003 shows lag for thread 3 of coordinated Replicat fin.

GLOBAL

Displays the lags in the GG_LAGS view.

Examples

Example 1

LAG REPLICAT *

Example 2

LAG REPLICAT *fin*

------------------------------------------------------------------------
REGISTER REPLICAT

Use the REGISTER REPLICAT command to register a Replicat group with a
target Oracle Database to support integrated Replicat mode. This command
should not be necessary under normal Replicat conditions. The startup
registers Replicat with the target database automatically. Use this
command only if Oracle GoldenGate returns a message that an integrated
Replicat is not registered with the database.

Before issuing this command, issue the DBLOGIN command as the Replicat
database user with privileges granted through
dbms_goldengate_auth.grant_admin_privilege.

Syntax

REGISTER REPLICAT group_name DATABASE

group_name

The name of a Replicat group or a wildcard (*) to specify multiple groups. For
example, T* registers all Replicat groups whose names begin with T.

DATABASE

Required keyword to register with the target database. Creates a
database inbound server and associates it with the specified Replicat group.

Example

REGISTER REPLICAT sales DATABASE
------------------------------------------------------------------------
```

```
SEND REPLICAT

Use SEND REPLICAT to communicate with a starting or running Replicat process.
The request is processed as soon as Replicat is ready to accept commands
from users.

Syntax

SEND REPLICAT group_name[threadID],
{

CACHEMGR {CACHESTATS | CACHEQUEUES | CACHEVMUSAGE} |
DEPENDENCYINFO|DEPINFO [TXNCOUNT num] |
FORCESTOP |
GETLAG |
GETPARAMINFO [parameter_name] [FILE output_file] |
HANDLECOLLISIONS | NOHANDLECOLLISIONS [table_spec] |
INTEGRATEDPARAMS(parameter_specification) |
REPORT [HANDLECOLLISIONS [table_spec]] |
RESUME |
STATUS |
STOP |
TRACE[2] [DDLINCLUDE | DDL[ONLY]] file_name |
TRACE[2] OFF |
TRACE OFF file_name |
TRACEINIT |
THREADS (threadID threadID] ...] thread_range thread_range]
 ...])
}

group_name[threadID]

The name of the Replicat group or the name of a specific thread of a
coordinated Replicat, for example fin003. If the command is issued for a
specific thread, then an option that is used applies only to that thread.
As an alternative, you can issue SEND REPLICAT with the THREADS option
instead of including threadID with the group name. If Replicat is not
running, an error is returned.

CACHEMGR {CACHESTATS | CACHEQUEUES | CACHEVMUSAGE}

Returns statistics about the Oracle GoldenGate memory cache manager.

    CACHESTATS
    Returns all CACHEMGR statistics.

    CACHEQUEUES
    Returns statistics for the free queues only.

    CACHEVMUSAGE
    Returns statistics for the virtual memory usage.

DEPENDENCYINFO|DEPINFO [TXNCOUNT num]

Prints out information from PR transaction dependency graph.
First, it shows the transaction groups currently being executed and then
the transactions waiting on some other due to a dependency.

TXNCOUNT: Determines the number of waiting transactions to print.
Default value is 10 and maximum is 99.
```

```
FORCESTOP

Forces Replicat to stop, bypassing any notifications. This command will
roll back any active transaction and stop the process immediately. This
command applies to Replicat as a whole and cannot be used for a specific
Replicat thread.

GETLAG

Shows a true lag time between Replicat and the trail. Lag time is the
difference, in seconds, between the time that the last record was processed
by Replicat and the timestamp of the record in the trail. The results
are the same as LAG REPLICAT.

GETPARAMINFO [parameter_name] [FILE output_file]

Use GETPARAMINFO to query runtime parameter values of a running instance,
including Extract, Replicat, and Manager. You can query for a single
parameter or all parameters and send the output to the console or a text
file.

    parameter_name

    The default behavior is to display all parameters in use, meaning
    those parameters that have ever been queried by the application,
    parameters, and their current values. If you specify a particular
    parameter, then the output is filtered by that name.

    FILEoutput_file

    The name of the text file that your output is redirected to.

HANDLECOLLISIONS | NOHANDLECOLLISIONS [table_spec]

Control HANDLECOLLISIONS behavior. Instead of using this option, you can
specify the HANDLECOLLISIONS or NOHANDLECOLLISIONS parameter in the
Replicat parameter file. This command can be sent directly
to an individual thread by means of SEND REPLICAT group_name[threadID]
or you can use the THREADS option to send the command through the
coordinator thread to affect multiple threads.

    HANDLECOLLISIONS

    Use HANDLECOLLISIONS to enable automatic error handling when performing
    initial data loads while the source database is active. Make certain to
    disable HANDLECOLLISIONS, either by issuing SEND REPLICAT with the
    NOHANDLECOLLISIONS option or by removing the parameter from the parameter
    file, after the initial load is complete and online data changes have
    been applied to the target tables.

    Note:
    The message returned by SEND REPLICAT with HANDLECOLLISIONS, when issued
    for a specific Replicat thread, shows that the command set HANDLECOLLISIONS
    for all MAP statements, not only the one handled by the specified thread.
    This is a known issue. The command actually affects only the MAP statement
    that includes the specified thread.

    NOHANDLECOLLISIONS

    Turns off the HANDLECOLLISIONS parameter but does not remove it from the
    parameter file. To avoid enabling HANDLECOLLISIONS the next time Replicat
```

```
    starts, remove it from the parameter file.

    table_spec

    table_spec restricts HANDLECOLLISIONS or NOHANDLECOLLISIONS to a specific
    target table or a group of target tables specified with a standard
    wildcard (*).

INTEGRATEDPARAMS(parameter_specification)
(Oracle) Supports an integrated Replicat. Sends a parameter specification
to the database inbound server while Replicat is running in integrated mode.
Only one parameter specification can be sent at a time with this command. You
can send multiple parameter changes, issue multiple SEND REPLICAT commands.

SEND REPLICAT myrep INTEGRATEDPARAMS ( parallelism 4 ) SEND REPLICAT
myrep INTEGRATEDPARAMS ( max_sga_size 250)

To preserve the continuity of processing, the parameter change is made
at a transaction boundary. For a list of supported inbound server
parameters, see the Oracle GoldenGate documentation for your database.

REPORT [HANDLECOLLISIONS [table_spec]]

Generates an interim statistical report to the Extract report file.
The statistics that are displayed depend upon the configuration of the
STATOPTIONS parameter when used with the RESETREPORTSTATS | NORESETREPORTSTATS
option. See “STATOPTIONS”.

    HANDLECOLLISIONS
    Shows tables for which HANDLECOLLISIONShas been enabled.

    table spec
    Restricts the output to a specific target table or a group of target
    tables specified with a standard wildcard (*).

RESUME

Resumes (makes active) a process that was suspended by an EVENTACTIONS SUSPEND
event. The process resumes normal processing from the point at which it was
suspended.

STATUS

Returns the current location within the trail and information regarding the
current transaction. Fields output are:

* Processing status (per thread, if Replicat is coordinated)

* Position in the trail file (per thread, if Replicat is coordinated)

* Trail sequence number (per thread, if Replicat is coordinated)

* RBA in trail

* Trail name

Possible processing status messages are:

* Delaying– waiting for more data

* Suspended– waiting to be resumed
```

```
* Waiting on deferred apply- delaying processing based on the
DEFERAPPLYINTERVAL parameter.

* Processing data- processing data

* Skipping current transaction- START REPLICAT with SKIPTRANSACTION was used.

* Searching for START ATCSN csn- START REPLICAT with ATCSN was used.

* Searching for START AFTERCSN csn- START REPLICAT with AFTERCSN was used.

* Performing transaction timeout recovery- stoping current incomplete
transaction and repositioning to start new one (see the TRANSACTIONTIMEOUT
parameter).

* Waiting for data at logical EOF after transaction timeout recovery-
Waiting to receive remainder of incomplete source transaction after a
TRANSACTIONTIMEOUT termination.

* At EOF (end of file)- no more records to process

Possible thread status messages when THREADS is used or the command is
issued for a specific thread are:

* Waiting for consensus stop point: This indicates that the threads are
 attempting to synchronize for a barrier transaction.

* Waiting for Watermark: Indicates that all threads are attempting to
stop at the same transaction boundary in the trail, known as the global
watermark.

* Waiting on all threads to start up: Indicates that the thread is
waiting for all of the threads to start after a successful barrier
transaction or a Replicat startup.

Possible coordinator thread status messages are:

* Waiting for all threads to register: Indicates that the MAP statements
are all being parsed to determine the thread IDs that are specified in them.

* Processing data: Indicates that data is being processed normally.

* Suspended, waiting to be resumed: Indicates that a SEND REPLICAT command
with a SUSPEND request was sent to Replicat.

* At EOF: Indicates that there is no more data in the trail to process.

* Waiting to register MAP statistics: Indicates that Replicat is collecting
processing statistics to send to the report file.

STOP

Stops Replicat gracefully. This command applies to Replicat as a whole and
cannot be used for a specific Replicat thread.

THREADS (threadID threadID] ...] thread_range
thread_range] ...])
Issues the command only for the specified thread or threads of a coordinated
Replicat. You can use this option or you can use groupname with threadID.
Without either of those options, the command applies to all active threads.
```

```
    threadID threadID] ...]
    Specifies a thread ID or a comma-delimited list of threads in the
    format of threadID, threadID, threadID.

    thread_range thread_range] ...]
    Specifies a range of threads in the form of threadIDlow-threadIDhigh
    or a comma-delimted list of ranges in the format of threadIDlow-threadIDhigh,
    threadIDlow-threadIDhigh.

A combination of these formats is permitted, such as threadID, threadID,
threadIDlow-threadIDhigh.

TRACE[2] [DDLINCLUDE | DDL[ONLY]] file_name
Turns tracing on and off. Tracing captures information to the specified
file to reveal processing bottlenecks. Tracing also can be enabled by
means of the Replicat parameters TRACE and TRACE2.

If the Replicat is in coordinated mode and TRACE is used with a THREADS
list or range, a trace file is created for each currently active thread.
Each file name is appended with its associated thread ID. This method of
identifying trace files by thread ID does not apply when SEND REPLICAT is
issued by groupname with threadID (as in SEND REPLICAT fin003TRACE...) or
when only one thread is specified with THREADS.

Contact Oracle Support for assistance if the trace reveals significant
processing bottlenecks.

    TRACE

    Captures step-by-step processing information.

    TRACE2

    Identifies code segments rather than specific steps.

    DDLINCLUDE | DDLONLY
    (Replicat only) Enables DDL tracing and specifies how DDL tracing
    is included in the trace report.

    * DDLINCLUDE includes DDL tracing in addition to regular tracing of
    transactional data processing.

    * DDL[ONLY] excludes the tracing of transactional data processing and
    only traces DDL. This option can be abbreviated to DDL.


    file_name

    file_name specifies the relative or fully qualified name of a file
    to which Oracle GoldenGate logs the trace information. If a trace is
    already in progress, the existing trace file is closed and the trace
    resumes to the file specified with file_name. For example:

    SEND REPLICAT group_name TRACE file_name DDLINCLUDE

    If no other options will follow the file name, the FILE keyword can be
    omitted, for example:

    SEND REPLICAT group_name TRACE DDLINCLUDE file_name
```

```
TRACE[2] OFF

Turns off tracing.

TRACE OFF file_name

Turns tracing off only for the specified trace file. This option supports
the EVENTACTIONS feature, where there can be multiple trace files due to
multiple EVENTACTIONS statements.

TRACEINIT

Resets tracing statistics back to 0 and then starts accumulating statistics
again. Use this option to track the current behavior of processing, as
opposed to historical.

Examples

Example 1

SEND REPLICAT finance, HANDLECOLLISIONS

Example 2

SEND REPLICAT finance, REPORT HANDLECOLLISIONS fin_*

Example 3

SEND REPLICAT finance, GETLAG

Example 4

SEND REPLICAT finance, INTEGRATEDPARAMS (parallelism 10)

Example 5

The following gets lag for thread 3 of a coordinated Replicat.

SEND REPLICAT fin003, GETLAG

Example 6

The following enables tracing for only thread 1 of a coordinated Replicat.
In this case, because only one thread is being traced, the trace file
will not have a threadID extension. The file name is trace.trc.

SEND REPLICAT fin, TRACE THREADS(1) FILE ./dirrpt/trace.trc

Example 7

The following enables tracing for threads 1,2, and 3 of a coordinated
Replicat. Assuming all threads are active, the tracing produces files
trace001, trace002, and trace003.

SEND REPLICAT fin TRACE THREADS(1-3) FILE ./dirrpt/trace.trc

Example 8

The following enables tracing only for thread 1 of a coordinated Replicat.
Because the command was issued directly for thread 1 without the use of a
THREAD clause, the trace file is named trace(without a thread ID suffix).
```

```
SEND REPLICAT fin001 TRACE FILE ./dirrpt/trace.trc

Example 9

The following sends dependency information to a Replicat.

Send Rep2 DEPENDENCYINFO TXNCOUNT 5

Sending DEPENDENCYINFO request to REPLICAT REP2 ...
Scheduler 0:
Transaction groups currently being executed:
    Group 0:0.3.31.2374, 0.1.15.1860, 0.6.19.2650
    Group 1:0.2.18.2510, 0.6.0.2645, 0.8.22.2487
    Group 2:0.9.7.2445, 0.10.6.1896, 0.3.8.2367
    Group 3:0.5.4.2362, 0.4.31.1736, 0.10.29.1875
    Group 4:0.3.6.2376, 0.6.2.2637, 0.9.8.2414
    Group 5:0.6.11.2645, 0.9.10.2451, 0.5.22.2357
    Group 6:0.9.20.2452, 0.3.4.2367, 0.6.20.2644
    Group 7:0.2.10.2487, 0.8.30.2459, 0.5.5.2360
    Group 8:0.2.2.2514, 0.8.4.2477, 0.4.23.1733
    Group 9:0.9.26.2454, 0.3.20.2367, 0.2.1.2483
Waiting transactions:
    Transaction with XID 0.8.28.2483 is waiting on transaction with XID
0.6.15.2617
    Transaction with XID 0.9.16.2453 is waiting on transaction with XID
0.8.28.2483
    Transaction with XID 0.7.19.1750 is waiting on transaction with XID
0.3.12.2377
    Transaction with XID 0.6.17.2647 is waiting on transaction with XID
0.7.19.1750
    Transaction with XID 0.9.18.2446 is waiting on transaction with XID
0.4.14.1732


------------------------------------------------------------------------
START REPLICAT

Use START REPLICAT to start Replicat. To confirm that Replicat has started,
use the INFO REPLICAT or STATUS REPLICAT command.

When starting an integrated Replicat group for an Oracle target database,
START REPLICAT automatically registers Replicat with the target database.

A coordinated Replicat can only be started as a whole. There is no option
to start individual threads. If the prior shutdown of a coordinated Replicat
was not clean, the threads may have stopped at different positions in the
trail file. If this happens, START REPLICAT writes a warning if the parameter
file was changed since the prior run and raises an error if the number of
threads was changed. To resolve these problems and start Replicat again,
see Administering Oracle GoldenGate.

Normal Start Point

Replicat can be started at its normal start point (from initial or current
checkpoints) or from an alternate, user-specified position in the trail.

START REPLICAT, without any options, causes Replicat to start processing
at one of the following points to maintain data integrity:

* After graceful or abnormal termination: At the first unprocessed transaction
```

in the trail from the previous run, as represented by the current read checkpoint.

* First-time startup after the group was created: From the beginning of the active trail file (seqno 0, rba 0). See Administering Oracle GoldenGate for more information about checkpoints.

Alternate Start Point

The SKIPTRANSACTION, ATCSN, and AFTERCSN options of START REPLICAT cause Replicat as a whole, or specific threads of a coordinated Replicat, to begin processing at a transaction in the trail other than the normal start point. Use these options to:

* Specify a logical recovery position when an error prevents Replicat from moving forward in the trail. Replicat can be positioned to skip the offending transaction or transactions, with the understanding that the data will not be applied to the target.

* Skip replicated transactions that will cause duplicate-record and missing-record
errors after a backup is applied to the target during an initial load. These options cause Replicat to discard transactions that occurred earlier than the most recent set of changes that were captured in the backup. You can map the value of the serial identifier that corresponds to the completion of the backup to a CSN value, and then start Replicat to begin applying transactions from the specified CSN onward.

Note:
Skipping a transaction, or starting at or after a CSN, might cause Replicat to start more slowly than normal, depending on how much data in the trail must be read before arriving at the appropriate transaction record. To view the startup progress, use the SEND REPLICAT command with the STATUS option. To omit the need for Replicat to read through transactions that ultimately will be skipped, you can use the ATCSN or AFTERCSN option when starting Extract and the data pumps, so that those transactions are omitted from the trail. See “START EXTRACT”. See Administering Oracle GoldenGate for more information about performing an initial load.

Syntax

```
START REPLICAT group_name
[SKIPTRANSACTION | {ATCSN csn | AFTERCSN csn}]
[FILTERDUPTRANSACTIONS | NOFILTERDUPTRANSACTIONS]
[THREADS (threadID threadID] ...] thread_range thread_range] ...])
```

group_name

The name of a Replicat group or a wildcard (*) to specify multiple groups. For example, T* starts all Replicat groups whose names begin with T.

SKIPTRANSACTION

Causes Replicat to skip the first transaction after its expected startup position in the trail. All operations from that first transaction are excluded.

If the MAXTRANSOPS parameter is also being used for this Replicat, it is possible
that the process will start to read the trail file from somewhere in the middle of a transaction. In that case, the remainder of the partial transaction is

skipped
and Replicat resumes normal processing from the next begin-transaction record in the file. The skipped records are written to the discard file if the DISCARDFILE
parameter is being used; otherwise, a message is written to the report file that is similar to:

User requested START SKIPTRANSACTION. The current transaction will be skipped. Transaction ID txid, position Seqno seqno, RBA rba

SKIPTRANSACTION is valid only when the trail that Replicat is reading is part of an online change synchronization configuration (with checkpoints). Not valid for task-type initial loads (where SPECIALRUN is used with ADD REPLICAT).

ATCSN csn| AFTERCSNcsn
Sets a user-defined start point at a specific CSN. When ATCSN or AFTERCSN is used, a message similar to one of the following is written to the report file:

User requested start at commit sequence number (CSN) csn-string

User requested start after commit sequence number (CSN) csn-string

General information about these options:

* Valid only when the trail that Replicat is reading is part of an online change synchronization configuration (with checkpoints). Not valid for task-type initial
loads (where SPECIALRUN is used with ADD REPLICAT).

* To support starting at, or after, a CSN, the trail must be of Oracle GoldenGate
version 10.0.0 or later, because the CSN is stored in the first trail record of each transaction. If Replicat is started with AFTERCSN against an earlier trail version, Replicat will abend and write an error to the report stating that the trail format is not supported.

ATCSN

Causes Replicat to start processing at the transaction that has the specified
CSN. Any transactions in the trail that have CSN values that are less than the specified one are skipped.

AFTERCSN

Causes Replicat to start processing at the transaction that occurred after the one with the specified CSN. Any transactions in the trail that have CSN values that are less than, or equal to, the specified one are skipped.

csn

Specifies a CSN value. Enter the CSN value in the format that is valid for the database. See Administering Oracle GoldenGate CSN formats and descriptions.
Replicat abends if the format is invalid and writes a message to the report file.
To determine the CSN to supply after an initial load is complete, use the commit
identifier at which the load utility completed the load. Otherwise, follow the
instructions in the initial load procedure for determining when to start

```
Replicat.

FILTERDUPTRANSACTIONS | NOFILTERDUPTRANSACTIONS

Causes Replicat to ignore transactions that it has already processed. Use when
Extract was repositioned to a new start point (see the ATCSN or AFTERCSN option
of
“START EXTRACT”) and you are confident that there are duplicate transactions
in the trail that could cause Replicat to abend. This option requires the use
of a checkpoint table. For Oracle Database, this option is valid only
for Replicat in nonintegrated mode. For Integrated mode and automatic
target trail file regeneration, the Integrated mode handles the duplicate
transactions transparently. The default is FILTERDUPTRANSACTIONS.

THREADS (threadID threadID] ...] thread_range
thread_range] ...])
Valid for SKIPTRANSACTION, ATCSN, and AFTERCSN when Replicat is in coordinated
mode. Not valid for START REPLICAT without those options. Starts the specified
Replicat thread or threads at the specified location.

    threadID threadID] ...]

    Specifies a thread ID or a comma-delimited list of threads in the format
    of threadID, threadID, threadID.

    thread_range thread_range] ...]

    Specifies a range of threads in the form of threadIDlow-threadIDhigh or
    a comma-delimted list of ranges in the format of threadIDlowthreadIDhigh,
    threadIDlow-threadIDhigh.

A combination of these formats is permitted, such as threadID, threadID,
threadIDlow-threadIDhigh.

Examples

Example 1

START REPLICAT finance

Example 2

The following starts Replicat at an Oracle-specific CSN.

START REPLICAT finance, ATCSN 6488359


Example 3

The following causes threads 4 and 5 of a coordinated Replicat to skip
the first transaction after their last checkpoint when Replicat is started.
If this were a 10-thread coordinated Replicat, threads 0-3 and 6-10 would
all start at the normal start point, that of their last checkpoint.

START REPLICAT fin SKIPTRANSACTION THREADS(4-5)

Example 4

The following example causes threads 1-3 of a coordinated Replicat to
start at CSN 6488359, threads 9-10 to start after CSN 6488360, and threads
7 and 8 to skip the first transaction after its last checkpoint.
```

```
START REPLICAT fin ATCSN 6488359 THREADS(1-3), AFTERCSN 6488360
THREADS(9-10), SKIPTRANSACTION THREADS(7,8)

------------------------------------------------------------------
STATS REPLICAT

Use STATS REPLICAT to display statistics for one or more Replicat groups.
Thread statistics for a coordinated Replicat group are provided as follows.

Thread Lag Gap
The difference between the maximum lag and the minimum lag among all threads.

Coordinated Total DDLs
The total number of coordinated DDL transactions.

Coordinated Total PK-Update Transactions
The total number of coordinated transactions that involved an update to a
primary key.

Coordinated Total EMI Transactions
The total number of coordinated EVENTACTIONS events.

Total Transactions with User-requested Coordination
The total number of coordinations that were explicitly requested in the
configuration by means of the COORDINATED option of the MAP parameter.

Average Coordination Time
The average time (in seconds) spent in coordination among all threads.

Syntax

STATS REPLICAT group_name
 [statistic]
 [TABLE [container. | catalog.]schema.table]
 [TOTALSONLY [container. | catalog.]schema.table]
 [REPORTCDR]
 [REPORTCHARCONV]
 [REPORTDETAIL | NOREPORTDETAIL]  REPORTRATE {HR | MIN | SEC}]
 ... ]

group_name

The name of a Replicat group or a wildcard (*) to specify multiple groups.
For example, T* shows statistics for all Replicat groups whose names begin with
T.

statistic

The statistic to be displayed. More than one statistic can be specified
by separating each with a comma, for example STATS REPLICAT finance, TOTAL,
DAILY.

Valid values are:

    TOTAL
    Displays totals since process startup.

    DAILY
    Displays totals since the start of the current day.
```

```
    HOURLY
    Displays totals since the start of the current hour.

    LATEST
    Displays totals since the last RESET command.

    RESET
    Resets the counters in the LATEST statistical field.

TABLE [container. | catalog.]schema.table]

Displays statistics only for the specified table or a group of tables
specified with a wildcard (*). The table name or wildcard specification
must be fully qualified with the two-part or three-part name, for example
hr.empor *.*.*.

TOTALSONLY [container. | catalog.]schema.table]

Summarizes the statistics for the specified table or a group of tables
specified with a wildcard (*). The table name or wildcard specification
must be fully qualified with the two-part or three-part name, for example
hr.empor *.*.*.

REPORTCDR

Shows statistics for Conflict Detection and Resolution. Statistics include:

* Total CDR conflicts

* CDR resolutions succeeded

* CDR resolutions failed

* CDR INSERTROWEXISTS conflicts

* CDR UPDATEROWEXISTS conflicts

* CDR UPDATEROWMISSING conflicts

* CDR DELETEROWEXISTS conflicts

* CDR DELETEROWMISSING conflicts


REPORTCHARCONV

Reports statistics for character validation when character-set conversion
is performed. The following statistics are added to the STATS output:

Total column character set conversion failure: the number of validation
or conversion failures in the current Replicat run.

Total column data truncation: the number of times that column data was
truncated in the current Replicat run as the result of character set conversion.

REPORTDETAIL | NOREPORTDETAIL

Controls whether or not the output includes operations that were not replicated
as the result of collision errors. These operations are reported in the
regular statistics (inserts, updates, and deletes performed) plus as
statistics in the detail display, if enabled. For example, if 10 records
```

were insert operations and they were all ignored due to duplicate keys, the report would indicate that there were 10 inserts and also 10 discards due to collisions. The default is REPORTDETAIL. See “STATOPTIONS”.

REPORTRATE {HR | MIN | SEC}

Displays statistics in terms of processing rate rather than absolute values.

HR
Sets the processing rate in terms of hours.

MIN
Sets the processing rate in terms of minutes.

SEC
Sets the processing rate in terms of seconds.

Examples

Example 1

The following example displays total and hourly statistics per minute for a specific
table, and it also resets the latest statistics. Statistics for discarded operations
are not reported.

STATS REPLICAT finance, TOTAL, HOURLY, TABLE sales.acct, REPORTRATE MIN, RESET, NOREPORTDETAIL

Example 2

The following example displays the same statistics as the previous example, but for thread 3 of a coordinated Replicat group.

STATS REPLICAT fin003, TOTAL, HOURLY, TABLE sales.acct, REPORTRATE MIN, RESET, NOREPORTDETAIL

STATUS REPLICAT

Use STATUS REPLICAT to determine whether or not Replicat is running. There are the following four possible statuses:

Abended

The process has abnormally ended.

Running

Means one of the following:

* Active: Running and processing (or able to process) data. This is the normal state of a process after it is started.

* Suspended: The process is running though suspended due to an EVENTACTIONS SUSPEND
action. In a suspended state, the process is not active, and no data can be processed, but the state of the current run is preserved and can be continued by issuing the RESUME command in GGSCI. The RBA in the INFO command reflects the last checkpointed position before the suspend action. To determine whether the

state is active or suspended, issue a SEND EXTRACT|REPLICAT group_name STATUS command. For more information, see SEND EXTRACT or SEND REPLICAT.

Starting

The process is starting.

Stopped

The process was stopped.

Syntax

```
STATUS REPLICAT group_name
 [TASKS]
 [ALLPROCESSES]
```

group_name

The name of a Replicat group or a wildcard (*) to specify multiple groups. For example, T* shows status for all Replicat groups whose names begin with T.

TASKS

Displays status only for Replicat tasks. By default, tasks are not displayed unless you specify a single Replicat group (without wildcards).

ALLPROCESSES

Displays status for all Replicat groups, including tasks.

Examples

Example 1

```
STATUS REPLICAT finance
```

Example 2

```
STATUS REPLICAT fin*
```

STOP REPLICAT

Use STOP REPLICAT to stop Replicat cleanly. This command preserves the state of synchronization for the next time Replicat starts, and it ensures that Manager does not automatically start Replicat.

In a clean shutdown of a coordinated Replicat, the coordinator thread attempts to stop all of the threads on the same transaction boundary. If the shutdown of a coordinated Replicat is not clean, the threads may stop at different positions in the trail file. If this happens, START REPLICAT writes a warning if the parameter file was changed since the prior run and raises an error if the number of threads was changed. To resolve these problems and start Replicat again, see Administering Oracle GoldenGate.

Syntax

```
STOP REPLICAT group_name [!]
```

group_name

The name of a Replicat group or a wildcard (*) to specify multiple groups. For example, T* stops all Replicat groups whose names begin with T.

!
(Exclamation point) Stops Replicat immediately. The transaction is stoped and the process terminates.

Example

```
STOP REPLICAT finance
```

SYNCHRONIZE REPLICAT

Use SYNCHRONIZE REPLICAT to return all of the threads of a coordinated Replicat to the same position in the trail file after an unclean shutdown. This position is the maximum checkpoint position of all of the threads, in other words, the most recent trail record processed among all of the threads. When SYNCHRONIZE REPLICAT is issued, all threads are started and allowed to process transactions until they reach the maximum checkpoint position, and then Replicat stops.

For more information about how to use SYNCHRONIZE REPLICAT to recover a coordinated Replicat after an unclean shutdown, or to enable repartitioning of data among different threads, see Administering Oracle GoldenGate.

Syntax

```
SYNCHRONIZE REPLICAT group_name
```

group_name

The name of a Replicat group or a wildcard (*) to specify multiple groups. For example, T* synchronizes the threads of all Replicat groups whose names begin with T. The threads synchronize to the same position within their group, not to the same position across all Replicat groups being synchronized with this command.

Example

```
SYNCHRONIZE REPLICAT repA
```

UNREGISTER REPLICAT

Use the UNREGISTER REPLICAT command to unregister a Replicat group from a target Oracle Database to disable integrated Replicat mode. Use this command only if you forcibly deleted the Replicat group. UNREGISTER REPLICAT should not be used when deleting Replicat in the normal manner, where you first stop Replicat and then issue the DELETE REPLICAT command.

Before issuing this command, issue the DBLOGIN command as the Replicat database user with privileges granted through dbms_goldengate_auth.grant_admin_privilege.

For more information about integrated Replicat, see the Oracle GoldenGate documentation
for your database.

Syntax

```
UNREGISTER REPLICAT group_name DATABASE

group_name

The name of a Replicat group or a wildcard (*) to specify multiple groups.
For example, T* unregisters all Replicat groups whose names begin with T.

DATABASE

Required keyword to unregister from the target database. Removes the database
inbound server that is associated with this Replicat.

Example

UNREGISTER REPLICAT sales DATABASE

------------------------------------------------------------------
SET_INSTANTIATION_CSN

SET_INSTANTIATION_CSN

Use SET_INSTANTIATION_CSN on your target database to set the instantiation CSN
manually. This command requires DBLOGIN. It enables a Replicat with the DBOPTIONS
ENABLE_INSTANTIATION_FILTERING option to filter out records below the specified
CSN for any object without Oracle Datapump import instantiation information.
It is an alternative to specifying @FILTER (@GETENV('TRANSACTION','CSN').

To enable instantiation SCN filtering, you must do the following:

1. Your Replicat parameter file must contain DBOPTIONS
ENABLE_INSTANTIATION_FILTERING.

2. The instantiation SCNs must be set at the target database for each table. You
can do this using one of the following two methods:
Automatically set the source SCN by the Oracle Datapump upon import if the
tables were prepared at the source database using ADD TRANDATA PREPARECSN or
ADD SCHEMATRANDATA PREPARECSN prior to the Oracle Datapump export.
or Manually set the instantiation source SCN at the target database using this
command.

Syntax

SET_INSTANTIATION_CSN csn FOR [schema.] table FROM source_database_name

csn

The CSN number that instantiation will begin.

[schema.]table

The name of the table to set the instantiation CSN on. If no schema is provided,
the DBLOGIN user will be used.

source_database_name

The global name of the source database for which this is a target.

Example

SET_INSTANTIATION_CSN 12345678 FOR hr.employees FROM DBS1.US.COMPANY.COM
```

```
------------------------------------------------------------------
CLEAR_INSTANTIATION_CSN

CLEAR_INSTANTIATION_CSN

Use CLEAR_INSTANTIATION_CSN on your target database to clear (reverse) the
instantiation CSN manually. This command requires DBLOGIN where the user is
the default Oracle GoldenGate schema.

Syntax

CLEAR_INSTANTIATION_CSN FOR [schema.]table FROM source_database_name

[schema.]table

The name of the table to clear the instantiation CSN on. If no schema is
provided, the DBLOGIN user will be used.

source_database_name

The global name of the source database for which this is a target.

Example

CLEAR_INSTANTIATION_CSN FOR hr.employees FROM DBS1.US.COMPANY.COM


------------------------------------------------------------------
####################################
#
#ER COMMANDS
#
#
####################################
------------------------------------------------------------------
INFO ER

INFO ER

Use the INFO ER command to get information on multiple Extract and
Replicat groups as a unit. Use it with wildcards to affect every
Extract and Replicat group that satisfies the wildcard. For
descriptions and optional parameters for this command, refer to the
INFO EXTRACT or INFO REPLICAT command.

Syntax:

INFO ER group wildcard specification


group wildcard specification

The wildcard specification for the groups that you want to affect with
the command. Oracle GoldenGate will automatically increase internal
storage to track up to 100,000 wildcard entries.


Example:

INFO ER *X*
```

```
------------------------------------------------------------------
KILL ER

KILL ER

Use the KILL ER command to forcefully terminate multiple Extract and
Replicat groups as a unit. Use it with wildcards to affect every
Extract and Replicat group that satisfies the wildcard. For
descriptions and optional parameters for this command, refer to the
KILL EXTRACT or KILL REPLICAT command.

Syntax:

KILL ER group wildcard specification


group wildcard specification

The wildcard specification for the groups that you want to affect with
the command. Oracle GoldenGate will automatically increase internal
storage to track up to 100,000 wildcard entries.


Example:

KILL ER *X*



------------------------------------------------------------------
LAG ER

LAG ER

Use the LAG ER command to get lag information on multiple Extract and
Replicat groups as a unit. Use it with wildcards to affect every
Extract and Replicat group that satisfies the wildcard. For
descriptions and optional parameters for this command, refer to the
LAG EXTRACT or LAG REPLICAT command.

Syntax:

LAG ER group wildcard specification


group wildcard specification

The wildcard specification for the groups that you want to affect with
the command. Oracle GoldenGate will automatically increase internal
storage to track up to 100,000 wildcard entries.


Example:

LAG ER *X*


------------------------------------------------------------------
SEND ER
```

```
SEND ER

Use the SEND ER command to send instructions to multiple Extract and
Replicat groups as a unit. Use it with wildcards to affect every
Extract and Replicat group that satisfies the wildcard. For
descriptions and optional parameters for this command, refer to the
SEND EXTRACT or SEND REPLICAT command.

Syntax:

SEND ER group wildcard specification


group wildcard specification

The wildcard specification for the groups that you want to affect with
the command. Oracle GoldenGate will automatically increase internal
storage to track up to 100,000 wildcard entries.


Example:

SEND ER *X*



------------------------------------------------------------------------
START ER

START ER

Use the START ER command to start multiple Extract and
Replicat groups as a unit. Use it with wildcards to affect every
Extract and Replicat group that satisfies the wildcard. For
descriptions and optional parameters for this command, refer to the
START EXTRACT or START REPLICAT command.

Syntax:

START ER group wildcard specification


group wildcard specification

The wildcard specification for the groups that you want to affect with
the command. Oracle GoldenGate will automatically increase internal
storage to track up to 100,000 wildcard entries.


Example:

START ER *X*



------------------------------------------------------------------------
STATS ER

STATS ER
```

Use the STATS ER command to get statistics on multiple Extract and Replicat groups as a unit. Use it with wildcards to affect every Extract and Replicat group that satisfies the wildcard. For descriptions and optional parameters for this command, refer to the STATS EXTRACT or STATS REPLICAT command.

Syntax:

STATS ER group wildcard specification

group wildcard specification

The wildcard specification for the groups that you want to affect with the command. Oracle GoldenGate will automatically increase internal storage to track up to 100,000 wildcard entries.

Example:

STATS ER *X*

STATUS ER

STATUS ER

Use the STATUS ER command to check the status of multiple Extract and Replicat groups as a unit. Use it with wildcards to affect every Extract and Replicat group that satisfies the wildcard. For descriptions and optional parameters for this command, refer to the STATUS EXTRACT or STATUS REPLICAT command.

Syntax:

STATUS ER group wildcard specification

group wildcard specification

The wildcard specification for the groups that you want to affect with the command. Oracle GoldenGate will automatically increase internal storage to track up to 100,000 wildcard entries.

Example:

STATUS ER *X*

STOP ER

STOP ER

Use the STOP ER command to stop multiple Extract and Replicat groups as a unit. Use it with wildcards to affect every Extract and Replicat group that satisfies the wildcard. For descriptions and optional parameters for this command, refer to the

```
STOP EXTRACT or STOP REPLICAT command.

Syntax:

STOP ER group wildcard specification


group wildcard specification>

The wildcard specification for the groups that you want to affect with
the command. Oracle GoldenGate will automatically increase internal
storage to track up to 100,000 wildcard entries.


Example:

STOP ER *X*


------------------------------------------------------------------------

####################################
#
#WALLET COMMANDS
#
#
####################################
------------------------------------------------------------------------
CREATE WALLET

Use the CREATE WALLET command to create a master-key wallet. This wallet
stores the master key that is used by Oracle GoldenGate processes to
encrypt the encryption keys that secure data over the network and in trail
files and other Oracle GoldenGate files that store sensitive data.

This command creates an empty wallet that remains open for the duration
of the GGSCI session. The GGSCI console returns messages similar to the
following, indicating that the wallet is present and open.

Created wallet at location './dirwlt'.

Opened wallet at location './dirwlt'.

The wallet is created as an autologinwallet (file extension .sso) to support
automated restarts of Oracle GoldenGate processes without requiring human
intervention to supply the necessary decryption passwords. The wallet file
is created in the directory specified by the GLOBAL parameter WALLETLOCATION,
if present, or otherwise in the default location of dirwltin the Oracle
GoldenGate installation directory.

The wallet is in a platform-independent format. It must either be stored on
a shared file system that is accessible by all systems in the Oracle GoldenGate
environment, or it must be copied to all of those systems initially and every
time the master key changes.

The wallet is permanent within Oracle GoldenGate, but can be manually deleted
with the appropriate command in the operating system, if that becomes necessary.

The use of a wallet and master key is not supported for the iSeries, z/OS,
and NonStop platforms.
```

See “ADD MASTERKEY”to add a master key value to the wallet.

Syntax

```
CREATE WALLET
```

OPEN WALLET

Use the OPEN WALLET command to open a master-key wallet. Opening a wallet decrypts the contents and loads them into the GGSCI memory. This command must be used before using any of the commands that add, renew, or delete the master keys in the wallet.

The wallet remains open for the rest of the GGSCI session. The name of the wallet to be opened is taken from the GLOBALS parameter WALLETLOCATION, if present, or otherwise it is opened from the default location in the Oracle GoldenGate installation directory.

The use of a wallet and master key is not supported for the iSeries, z/OS, and NonStop platforms.

Syntax

```
OPEN WALLET
```

PURGE WALLET

Use the PURGE WALLET command to permanently remove master key versions from the master-key wallet. Only the versions that are marked for deletion by the DELETE MASTERKEY command are removed. The purge is not reversible.

Note:
For Oracle GoldenGate deployments using a shared wallet, the older versions of the master key should be retained after the master key is renewed until all processes are using the newest version. The time to wait depends on the topology, latency, and data load of the deployment. A minimum wait of 24 hours is a conservative estimate, but you may need to perform testing to determine how long it takes for all processes to start using a new key.To determine whether all of the processes are using the newest version, view the report file of each Extract immediately after renewing the master key to confirm the last SCN that was mined with the old key. Then, monitor the Replicat report files to verify that this SCN was applied by all Replicat groups. At this point, you can delete the older versions of the master key.

The OPEN WALLET command must be used before using this command or any of the commands that add, renew, or delete the master keys in the wallet.

After purging a wallet that is not maintained centrally on shared storage, the updated wallet can be copied to all of the other systems in the Oracle GoldenGate configuration that use this wallet, so that no purged keys remain in the configuration. Before doing so, Extract must be stopped and then all of the downstream Oracle GoldenGate processes must be allowed to finish processing their trails and then be stopped. After the wallet is copied into place, the processes can be started again. For detailed instructions, see Administering Oracle GoldenGate.

The use of a wallet and master key is not supported for the iSeries,

```
z/OS, and NonStop platforms.

Syntax
PURGE WALLET


------------------------------------------------------------------
###################################
#
#MASTERKEY COMMANDS
#
#
###################################
------------------------------------------------------------------
ADD MASTERKEY

Use the ADD MASTERKEY command to add a master key to a master-key wallet.
The master key is used by Extract and Replicat to encrypt the encryption
keys that secure data being sent across the network and in the trail files,
so that those keys can be sent to, and used, by downstream processes. The
master key omits the need to use wallet storage for the keys that actually
encrypt the data.

The master-key wallet must be open to add a key. Use the CREATE WALLET
or OPEN WALLET command to open a wallet. The wallet remains open throughout
the same GGSCI session in which the command was issued.

The master key is generated as a random sequence of bits.

The successful completion of this command returns a message similar to
the following:
Created version 1 of master key  'OGG_DEFAULT_MASTERKEY' in Oracle Wallet
'dirwlt/cwallet.sso

After adding a master key to a wallet that is not maintained centrally on
shared storage, the updated wallet must be copied to all of the other
systems in the Oracle GoldenGate configuration that use this wallet.
Before doing so, Extract must be stopped and then all of the downstream
Oracle GoldenGate processes must be allowed to finish processing their
trails and then be stopped. After the wallet is copied into place, the
processes can be started again. For detailed instructions, see Administering
Oracle GoldenGate.

The use of a wallet and master key is not supported for the iSeries,
z/OS, and NonStop platforms.

Syntax

ADD MASTERKEY

Example

ADD MASTERKEY


------------------------------------------------------------------
INFO MASTERKEY

Use the INFO MASTERKEY command to view the contents of the master-key wallet.
The default output shows the version history of the master key, with the
creation date of a version and the status of the version. The status can be
```

```
one of the following:

* Current: Indicates this is the active version of the master key.

* Available: Indicates this version is not the current one, but can be
made active if needed.

* Deleted: Indicates that this version is marked to be deleted when the
PURGE WALLET command is issued.


The use of a wallet and master key is not supported for the iSeries,
z/OS, and NonStop platforms.

Syntax

INFO MASTERKEY [VERSION version]

VERSION version

Shows detailed information about a specific version of the master key.
The output includes the original creation date, the latest renewal date,
the status, and the hash of AES (Advanced Encryption Standard) Key.

Examples

Example 1

The following example shows the default input without any options.

INFO MASTERKEY

Masterkey Name:         OGG_DEFAULT_MASTERKEY
Creation Date:          Mon Aug 27 10:00:40 2017
Version:  Creation Date:            Status:
1         Mon Aug 27 10:00:40 2017  Deleted
2         Mon Aug 27 10:00:46 2017  Available
3         Mon Aug 27 10:02:58 2017  Deleted
4         Mon Aug 27 10:03:02 2017  Deleted
5         Mon Aug 27 10:03:05 2017  Deleted
6         Mon Aug 27 10:03:09 2017  Available
7         Mon Aug 27 10:03:16 2017  Current

Example 2

The following example shows the results of INFO MASTERKEY with VERSION.
The status of Current in the output shows that version 7 is the active version.

INFO MASTERKEY VERSION 7

Masterkey Name:     OGG_DEFAULT_MASTERKEY
Creation Date:      Mon Aug 27 10:00:40 2017
Version:         7
Renew Date:         Mon Aug 27 10:03:16 2017
Status:         Current
Key Hash (SHA1):    0xC65ADFA1CF42F9DB2CED3BC39A53F661CDED3304

------------------------------------------------------------------------
RENEW MASTERKEY

Use the RENEW MASTERKEY command to create a new version of the master
```

encryption key in the master-key wallet. The key name remains the same, but the bit ordering is different. All versions of a master key remain in the wallet until they are marked for deletion with the DELETE MASTERKEY command and then the wallet is purged with the PURGE WALLET command.

The OPEN WALLET command must be used before using this command or any of the commands that add or delete the master keys or purge the wallet.

A message similar to the following indicates that the command succeeded.

Masterkey 'OGG_DEFAULT_MASTERKEY' renewed to version 2 in wallet at location './ dirwlt'.

After renewing a master key in a wallet that is not maintained centrally on shared storage, the updated wallet must be copied to all of the other systems in the Oracle GoldenGate configuration that use this wallet. Before doing so, Extract must be stopped and then all of the downstream Oracle GoldenGate processes must be allowed to finish processing their trails and then be stopped. After the wallet is copied into place, the processes can be started again. For detailed instructions, see Administering Oracle GoldenGate.

The use of a wallet and master key is not supported for the iSeries, z/OS, and NonStop platforms.

Syntax

RENEW MASTERKEY

Example

This example creates a new version of the master key.

RENEW MASTERKEY

DELETE MASTERKEY

DELETE MASTERKEY

Use the DELETE MASTERKEY command to mark a version of a master key for deletion. Routinely deleting older versions of a master key ensures that they cannot be used maliciously.

The OPEN WALLET command must be used before using this command or any of the commands that add or renew the master keys or purge the wallet.

To view the version of a master key, use the INFO MASTERKEY command.

This command marks a version for deletion but does not physically remove it from the wallet. See “PURGE WALLET” to remove the master key version permanently.

Note:
For Oracle GoldenGate deployments using a shared wallet, the older versions of the master key should be retained after the master key is renewed until all processes are using the newest version. The time to wait depends on the topology, latency, and data load of the deployment. A minimum wait of 24 hours is a conservative estimate, but you may need to perform testing to determine how long it takes for all processes to start using a new key. To determine whether all of the processes are using the newest version, view the report file of

each Extract immediately after renewing the master key to confirm the last SCN that was mined with the old key. Then, monitor the Replicat report files to verify that this SCN was applied by all Replicat groups. At this point, you can delete the older versions of the master key.

See “UNDELETE MASTERKEY” to reverse a deletion made by DELETE MASTERKEY.

Once a version number is used, the wallet reserves it forever, and no other key of the same version can be generated. For example, you cannot mark version 2 of a key for deletion, then purge the wallet to remove it, and then issue RENEW MASTERKEY to add a version 2 again. Even though only version 1 of the key remains in the wallet after the purge, the renewal generates version 3, not version 2.

The use of a wallet and master key is not supported for the iSeries, z/OS, and NonStop platforms.

Syntax

```
DELETE MASTERKEY
{VERSION version | RANGE FROM begin_value TO end_value | ALL}
```

VERSION version

Specifies a single version to be marked for deletion.

RANGE FROM begin_value TO end_value

Specifies a range of versions to be marked for deletion. The versions must be contiguous. For example, specifying RANGE FROM 3 TO 6 marks versions 3, 4, 5, and 6.

ALL

Marks all versions of the master key for deletion, including the currently active one. When this option is used, it should always be followed by a RENEW MASTERKEY command to create a new, current version of the master key.

Examples

Example 1

This command marks one version of the master key for deletion and returns a message similar to the one shown.

```
DELETE MASTERKEY VERSION 10

Version 10 of Masterkey 'OGG_DEFAULT_MASTERKEY' deleted from wallet at
location './
dirwlt'.
```

Example 2

This command marks versions 3, 4, 5, and 6 for deletion and returns a message similar to the one shown.

```
DELETE MASTERKEY RANGE FROM 3 TO 6

Version 3 of Masterkey 'OGG_DEFAULT_MASTERKEY' deleted from wallet
at location './ dirwlt'.
```

```
Version 4 of Masterkey 'OGG_DEFAULT_MASTERKEY' deleted from wallet
at location './ dirwlt'.
Version 5 of Masterkey 'OGG_DEFAULT_MASTERKEY' deleted from wallet
at location './ dirwlt'.
Version 6 of Masterkey 'OGG_DEFAULT_MASTERKEY' deleted from wallet
at location './ dirwlt'.

------------------------------------------------------------------
UNDELETE MASTERKEY

Use the UNDELETE MASTERKEY command to remove the deletion mark from a
master key version, thus retaining that version if the PURGE WALLET
command is used. Only one version can be unmarked per UNDELETE MASTERKEY
command. See “DELETE MASTERKEY” to mark a version of a master key for
deletion.

The OPEN WALLET command must be used before using this command or any
of the commands that add, renew, or delete the master keys in the wallet.

The use of a wallet and master key is not supported for the iSeries,
z/OS, and NonStop platforms.

Syntax

UNDELETE MASTERKEY VERSION version

VERSION version

The version that is to be unmarked for deletion.

Example

This command unmarks version 3 of the master key and returns a message
similar to the one shown.

UNDELETE MASTERKEY VERSION 3
Version 3 of Masterkey 'OGG_DEFAULT_MASTERKEY' undeleted from wallet
at location './ dirwlt'.

------------------------------------------------------------------
####################################
#
#CREDENTIALSTORE COMMANDS
#
#
####################################
------------------------------------------------------------------
ADD CREDENTIALSTORE

Use the ADD CREDENTIALSTORE command to create a credential store. The
credential store manages user IDs and their encrypted passwords (together
known as credentials) that are used by Oracle GoldenGate processes to
interact with the local database. The credential store eliminates the
need to specify user names and clear-text passwords in the Oracle GoldenGate
parameter files. An optional alias can be used in the parameter file
instead of the user ID to map to a userid-password pair in the credential
store.

The credential store is implemented as an autologin wallet within the
Oracle Credential Store Framework (CSF). The use of an LDAP directory
is not supported for the Oracle GoldenGate credential store. The autologin
```

wallet supports automated restarts of Oracle GoldenGate processes without requiring human intervention to supply the necessary passwords.

ADD CREDENTIALSTORE creates an empty credentials store in the location that is specified with the CREDENTIALSTORELOCATION parameter in the GLOBALS file, if used, or otherwise in the default location of dircrdin the Oracle GoldenGate installation directory. A credential store can be shared by multiple instances (installations) of Oracle GoldenGate on the same or different systems. Store a shared credential store in a shared file system, and specify this location in each Oracle GoldenGate instance by using the CREDENTIALSTORELOCATION parameter in each GLOBALS parameter file.

Only one credential store can be used at a time by any given instance of Oracle GoldenGate. For example, you can have a credential store named /home/ogg/ credentialsand a credential store named /test/ogg/credentials, but only one can be used at runtime by a given instance of Oracle GoldenGate. You can stop the processes to switch to a different credential store, but make certain to update the CREDENTIALSTORELOCATION parameter in each GLOBALS parameter file, and change the USERIDALIAS parameters to specify different aliases if needed.

The use of a credential store is not supported for the iSeries, z/OS, and NonStop platforms.

For more information about Oracle GoldenGate security options, see Administering Oracle GoldenGate.

Syntax

```
ADD CREDENTIALSTORE
```

ALTER CREDENTIALSTORE

Use the ALTER CREDENTIALSTORE command to manage user ID and password pairs in the credential store. This command enables you to add credentials to the credential store and to specify different aliases for a user. Upon successful completion, the command returns a message similar to the following:

```
Credential store altered.
```

The use of a credential store is not supported for the DBE for i, DB2 z/OS, and NonStop platforms.

For more information about Oracle GoldenGate security options, see Administering Oracle GoldenGate.

Syntax

```
ALTER CREDENTIALSTORE {
 ADD USER userid |
 REPLACE USER userid |
 DELETE USER userid }
[PASSWORD password]
[ALIAS alias]
[DOMAIN domain]
```

ADD USER userid

Adds the specified user and its alias to the credential store. If the ALIAS option is not used, the alias defaults to the user name. A credential can

only be entered once unless the ALIAS option is used to specify a different alias for each one. Unless the PASSWORD option is used, the command prompts for the password of the specified user. The user can be an actual user name or a SQL*Net connect string.

REPLACE USER userid

Changes the password of the specified user. If the ALIAS option is not used, the alias defaults to the user name. You cannot change the alias or domain of a user with this option, but you can use the ADD USER option to add a new entry for the user under the desired ALIAS or DOMAIN. Unless the PASSWORD option is used, the command prompts for the new password for the specified user.

DELETE USER userid

Removes the credential for the specified user from the credential store. If the ALIAS option is not used, the alias defaults to the user name.

PASSWORD password

The user's password. The password is echoed (not obfuscated) when this option is used. If this option is omitted, the command prompts for the password, which is obfuscated as it is typed (recommended as more secure).

```
ALTER CREDENTIALSTORE ADD USER scott
Password: ********
```

ALIAS alias

Specifies an alias for the user name. Use this option if you do not want the user name to be in a parameter file or command. If ALIAS is not used, the alias defaults to the USER name, which then must be used in parameter files and commands where a login is required. You can create multiple entries for a user, each with a different alias, by using the ADD USER option with ALIAS.

DOMAIN domain

Saves the credential user under the specified domain name. Enables the same alias to be used by multiple Oracle GoldenGate installations that use the same credential store. The default domain is Oracle GoldenGate. For example, the administrators of system 1 might not want system 2 to have access to the same credentials that are used on system 1. Those credentials can be stored as ALIAS extract, for example, under DOMAIN system1, while a different set of credentials can be stored for ALIAS extractunder DOMAIN system2.

Examples

Example 1

This example adds a user named scott but omits the PASSWORD specification, so the command prompts for Scott's password.

```
ALTER CREDENTIALSTORE ADD USER scott
Password: ********
```

Example 2

This example adds the user scott with his password tiger and specifies an alias for scott that is named scsm2.

```
ALTER CREDENTIALSTORE ADD USER scott PASSWORD tiger ALIAS scsm2

Example 3

This example adds the user scott under the domain of support.

ALTER CREDENTIALSTORE ADD USER scott ALIAS scsm3 DOMAIN support
Password: ********

Example 4

This example issues two ALTER CREDENTIALSTORE commands, each of which
adds a scott entry, but with a different alias.

ALTER CREDENTIALSTORE ADD USER scott ALIAS scsm2
Password: ********
ALTER CREDENTIALSTORE ADD USER scott ALIAS scsm3
Password: ********

Example 5

The following shows how the DELETE USER option works with and without the
ALIAS option.

The following command deletes the user1 entry for which the ALIAS is the
same as the user name.

ALTER CREDENTIALSTORE DELETE USER user1

The following command deletes the entry for user user1 that is associated
with the alias alias1.

ALTER CREDENTIALSTORE DELETE USER user1 ALIAS alias1
Alias: alias1
Userid: user1

Example 6
This example uses a SQL*Net connect string as the user value. In this case,
the PASSWORD option is omitted. The person issuing the command will be
prompted for the password, which is obfuscated.
ALTER CREDENTIALSTORE ADD USER oggext1@ora1 ALIAS ora1

------------------------------------------------------------------------
INFO CREDENTIALSTORE

Use the INFO CREDENTIALSTORE command to get information about an Oracle
GoldenGate credential store. This information includes the aliases that a
credential store contains and the user IDs that correspond to them. The
encrypted passwords in the credential store are not returned.

The credential store location is identified by the CREDENTIALSTORELOCATION
parameter in the GLOBALS file, if one exists, or otherwise by the default
location of dircrdin the Oracle GoldenGate installation directory.

The use of a credential store is not supported for the iSeries, z/OS,
and NonStop platforms.

For more information about Oracle GoldenGate security options,
see Administering Oracle GoldenGate.

Syntax
```

```
INFO CREDENTIALSTORE [DOMAIN domain]

DOMAIN domain

Returns the aliases and user IDs for a specific domain. For security purposes,
if the DOMAIN option is omitted, only the aliases and user IDs under the
default domain of OracleGoldenGateare shown. It is not possible to see DOMAIN
credentials unless the person issuing the INFO CREDENTIALSTORE command knows
the name of the domain. See “ALTER CREDENTIALSTORE” for more information about
domains.

Examples

Example 1

The following example shows the default output of INFO CREDENTIALSTORE.

INFO CREDENTIALSTORE
Domain: OracleGoldenGate
 Alias: support1
 Userid: scott
 Alias: sales1
 Userid: scott

Example 2

The following example shows the output when DOMAINis used.
INFO CREDENTIALSTORE DOMAIN support
Domain: Support
 Alias: support1
 Userid: scott

------------------------------------------------------------------------
DELETE CREDENTIALSTORE

Use the DELETE CREDENTIALSTORE command to remove a credential store from
the system. The credential store wallet and its contents are permanently
deleted.

The use of a credential store is not supported for the DBE for i, DB2 z/OS, and
NonStop platforms

For more information about Oracle GoldenGate security options, see
Administering Oracle GoldenGate.

Syntax

DELETE CREDENTIALSTORE

------------------------------------------------------------------------

####################################
#
#TRAIL COMMANDS
#
#
####################################
------------------------------------------------------------------------

ADD EXTTRAIL
```

Use ADD EXTTRAIL to create a trail for online processing on the local system and:

* Associate it with an Extract group.

* Assign a maximum file size.

Syntax

```
ADD EXTTRAIL trail_name, EXTRACT group_name
 MEGABYTES n]
[SEQNO n]
```

trail_name

The relative or fully qualified path name of the trail. The trail name can contain only two characters. Oracle GoldenGate appends this name with a nine-digit sequence number whenever a new file is created. For example, a trail named dirdat/tr would have files named dirdat/tr000000001, dirdat/tr000000002, and so forth.

group_name

The name of the Extract group to which the trail is bound. Only one Extract process can write data to a trail.

MEGABYTES n

The maximum size, in megabytes, of a file in the trail. The default is 500.

SEQNO n

Specifies that the first file in the trail will start with the specified trail sequence number. Do not include any zero padding. For example, to start at sequence 3 of a trail named tr, specify SEQNO 3. The actual file would be named /ggs/dirdat/tr000000003. This option can be used during troubleshooting when Replicat needs to be repositioned to a certain trail sequence number. It eliminates the need to alter Replicat to read the required sequence number.

Examples

Example 1

```
ADD EXTTRAIL dirdat/aa, EXTRACT finance, MEGABYTES 200
```

Example 2

```
ADD EXTTRAIL /ggs/dirdat/tr000000003
```

ADD RMTTRAIL

Use ADD RMTTRAIL to create a trail for online processing on a remote system and:

* Assign a maximum file size.

* Associate the trail with an Extract group.

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

Note:
The RMTTRAIL size (Target Trail) must be greater than or equal to the EXTTRAIL size (Source Trail), due to trail encryption requirements.

Syntax:

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

trail_name

The relative or fully qualified path name of the trail. The actual trail name can contain only two characters. Oracle GoldenGate appends this name with a nine-digit sequence number whenever a new file is created. For example, a trail named ./ dirdat/tr would have files named ./dirdat/tr000000001, ./dirdat/tr000000002, and so forth.

group_name

The name of the Extract group to which the trail is bound. Only one primary Extract process can write data to a remote trail.

MEGABYTES n

The maximum size, in megabytes, of a file in the trail. The default is 500.

SEQNO n

Specifies that the first file in the trail will start with the specified trail sequence number. Do not include any zero padding. For example, to start at sequence 3 of a trail named tr, specify SEQNO 3. The actual file would be named /ggs/dirdat/tr000000003. This option can be used during troubleshooting when Replicat needs to be repositioned to a certain trail number. It eliminates the need to alter Replicat to read the required sequence number.

Example

Example 1

```
ADD RMTTRAIL dirdat/aa, EXTRACT finance, MEGABYTES 200
```

Example 2

```
ADD RMTTRAIL /ggs/dirdat/tr000000003
```

ALTER EXTTRAIL

Use ALTER EXTTRAIL to change the attributes of a trail that was created with the ADD EXTTRAIL command (a trail on the local system). The change takes effect the next time that Extract starts.

Before using this command, stop the Extract using the STOP EXTRACT group_name command.

Syntax

```
ALTER EXTTRAIL trail_name, EXTRACT group_name
 MEGABYTES n]
```

trail_name

The relative or fully qualified path name of the trail. For example, dirdat/aa.

group_name

The name of the Extract group to which the trail is bound.

MEGABYTES n

The maximum size of a file, in megabytes. The default is 500. After using this option, issue the SEND EXTRACT command with the ROLLOVER option to close the current trail file and open a new one.

Example

```
ALTER EXTTRAIL dirdat/aa, EXTRACT finance, MEGABYTES 200
```

ALTER RMTTRAIL

Use ALTER RMTTRAIL to change the attributes of a trail that was created with the ADD RMTTRAIL command (a trail on a remote system). The change takes effect the next time that Extract starts.

Syntax

```
ALTER RMTTRAIL trail_name, EXTRACT group_name
 MEGABYTES n]
```

trail_name

The relative or fully qualified path name of the trail. For example, dirdat/aa.

group_name

The name of the Extract group to which the trail is bound.

MEGABYTES n

The maximum size of a file, in megabytes. The default is 500. After using this option, issue the SEND EXTRACT command with the ROLLOVER option to close the current trail file and open a new one.

Example

```
ALTER RMTTRAIL dirdat/aa, EXTRACT finance, MEGABYTES 200
```

DELETE EXTTRAIL

Use DELETE EXTTRAIL to delete the record of checkpoints associated with a

trail on a local system. Checkpoints are maintained in a file bearing the same name as the group in the dirchk sub-directory of the Oracle GoldenGate directory.

This command only deletes references to the specified trail from the checkpoint file. It does not delete the trail files themselves. To delete the trail files, use the PURGE EXTTRAIL command. See Administering Oracle GoldenGate for more information about checkpoints.

Syntax

```
DELETE EXTTRAIL trail_name
```

trail_name

The relative or fully qualified path name of the trail, including the two-character trail prefix.

Example

```
DELETE EXTTRAIL dirdat/et
```

DELETE RMTTRAIL

Use DELETE RMTTRAIL to delete the record of checkpoints associated with a trail on a remote system. Checkpoints are maintained in a file bearing the same name as the group in the dirchk sub-directory of the Oracle GoldenGate directory.

This command only deletes references to the specified trail from the checkpoint file. It does not delete the trail files themselves.

Syntax

```
DELETE RMTTRAIL trail_name EXTRACT group_name
```

trail_name

The relative or fully qualified path name of the trail, including the two-character trail prefix.

group_name

The name of the Extract group to which the trail is bound. If not specified, DELETE RMTTRAIL deletes the trail reference from all Extract groups that write to the specified trail.

Example

```
DELETE RMTTRAIL dirdat/et
```

INFO EXTTRAIL

Use INFO EXTTRAIL to retrieve configuration information for a local trail. It shows the name of the trail, the Extract that writes to it, the position of the last data processed, and the assigned maximum file size.

Syntax

```
INFO EXTTRAIL trail_name

trail_name

The relative or fully qualified path name of the trail or a wildcard
designating multiple trails.

Examples

Example 1

INFO EXTTRAIL dirdat/aa

Example 2

INFO EXTTRAIL *

Example 3

The following is sample output of INFO EXTTRAIL.
Extract Trail: c:\gg_81\dirdat\md
 Extract: GGSEXT8
 Seqno: 2
 RBA: 51080
 File Size: 100M

------------------------------------------------------------------------
INFO RMTTRAIL

Use INFO RMTTRAIL to retrieve configuration information for a remote trail.
It shows the name of the trail, the Extract that writes to it, the position
of the last data processed, and the assigned maximum file size.

Syntax

INFO RMTTRAIL trail_name

trail_name

The relative or fully qualified path name of the trail or a wildcard
designating multiple trails.

Examples

Example 1

INFO RMTTRAIL dirdat/aa

Example 2

INFO RMTTRAIL *

Example 3

The following is a sample of INFO RMTTRAILoutput.
Extract Trail: /ogg/dirdat/aa
 Seqno Length: 9
 Flip Seqno Length: no
 Extract: OGGPMP
 Seqno: 4
 RBA: 78066
```

```
 File Size: 500M

------------------------------------------------------------------
VIEW PARAMS

Use VIEW PARAMS to view the contents of a parameter file.
    Caution:
    Do not use this command to view a parameter file that is in a character
    set other than that of the local operating system (such as one where
    the CHARSET option was used to specify a different character set). The
    contents may become corrupted. View the parameter file from outside GGSCI.

Syntax

VIEW PARAMS {MGR | group_name | file_name}

MGR

Shows the Manager parameter file.

group_name

Shows the parameter file for the specified Extract or Replicat group.

file_name

Shows the specified file. By default, the subdirectory dirprm is used if no
path is specified. If the parameter file resides in a directory other than
dirprm, specify the full path name.

Examples

Example 1

VIEW PARAMS finance

Example 2

VIEW PARAMS c:\lpparms\replp.prm

------------------------------------------------------------------
####################################
#
#PARAMETER COMMANDS
#
#
####################################
------------------------------------------------------------------
EDIT PARAMS

Use EDIT PARAMS to create or change a parameter file. By default, this
launches Notepad on Windows systems or the vi editor on UNIX systems. You
can change the editor with the SET EDITOR command.

    Caution:
    Do not use this command to view or edit an existing parameter file that
    is in a character set other than that of the local operating system (such
    as one where the CHARSET option was used to specify a different character
    set). The contents may become corrupted. View the parameter file from
    outside GGSCI.
```

Syntax

EDIT PARAMS {MGR | group_name | file_name}

MGR

Opens a parameter file for the Manager process.

group_name

Opens a parameter file for the specified Extract or Replicat group.

file_name

Opens the specified file. When you create a parameter file with EDIT PARAMS in GGSCI, it is saved to the dirprm sub-directory of the Oracle GoldenGate directory. You can create a parameter file in a directory other than dirprm by specifying the full path name, but you must also specify the full path name with the PARAMS option of the ADD EXTRACT or ADD REPLICAT command when you create the process group.

Examples

Example 1

EDIT PARAMS finance

Example 2

EDIT PARAMS c:\lpparms\replp.prm

SET EDITOR

Use SET EDITOR to change the default text editor for the current session of GGSCI. The default editors are Notepad for Windows and vi for UNIX. GGSCI input, including to create parameter files, takes the character set of the local operating system.

Syntax

SET EDITOR program_name

program_name

Any text editor.

Example

The following example changes the default editor to Wordpad.

SET EDITOR wordpad

INFO PARAM

Use INFO PARAM to retrieve the parameter definition information. If a name matches multiple records, they are all displayed. If the query parameter has child options, they are not displayed in the output though their names

are listed in the Options tab. To display the full record of an option, the full name in the form of parameter.option should be queried separately.

This parameter infrastructure allows unlimited levels of options. Thus, a full name of a parameter or option might have numbers of segments, such as A.B.C.D.

Syntax

INFO PARAM name

name
The name of a parameter, an option, or a full name that is part of the several names concatenated together using dot ('.') as the delimiter. These sample names are valid:

* STREAMING

* RMTHOST.STREAMING

* RMTHOST

* RMTHOSTOPTIONS.STREAMING

* TRANLOGOPTIONS.INTEGRATEDPARAM.EAGER_SIZE

The matching with this set of sample names is that STREAMING matches as an option of both RMTHOST and RMTHOSTOPTIONS.

Example

INFO PARAM RMTHOST

GETPARAMINFO

Use GETPARAMINFO to query runtime parameter values of a running instance, including Extract, Replicat, and Manager. You can query for a single parameter or all parameters and send the output to the console or a text file.

Syntax

SEND MGR | group GETPARAMINFO [parameter_name] [FILE output_file]

group

The name of the Extract or Replicat instance or MGR.

parameter_name

The default behavior is to display all parameters in use, meaning those parameters that have ever been queried by the application, parameters, and their current values. If you specify a particular parameter, then the output is filtered by that name.

FILEoutput_file

The name of the text file that your output is redirected to.

Examples

```
Example 1

This example displays one parameter.

SEND MGR GETPARAMINFO PORT

Example 2

This example displays all parameters loaded from parameter file into Replicat
rep1 and those parameters that the rep1 has accessed.

SEND REPL GETPARAMINFO

Example 3

The following example redirects the output to a file.

SEND MGR GETPARAMINFO FILE mgrfile.out

------------------------------------------------------------------------
###################################
#
#DATABASE COMMANDS
#
#
###################################
------------------------------------------------------------------------
DBLOGIN

Use DBLOGIN to establish a database connection through GGSCI in preparation
to issue other Oracle GoldenGate commands that affect the database. The user
who issues DBLOGIN should have the appropriate database privileges to perform
the functions that are enacted by those commands. Any other special privileges
that are required for a GGSCI command are listed with the reference
documentation
for that command.

Requirements When Configuring Extract or Replicat in Integrated Mode (Oracle)

If using DBLOGIN to issue ADD EXTRACT, ALTER EXTRACT, or REGISTER EXTRACT to
initiate integrated capture or ADD REPLICAT, ALTER REPLICAT, or REGISTER REPLICAT
to initiate integrated Replicat against an Oracle Database, the user who issues
DBLOGIN must:

* Have privileges granted through the Oracle
dbms_goldengate_auth.grant_admin_privilege
procedure.

* Not be changed while Extract or Replicat is in integrated mode.


Special Database Privileges to Use Log Retention in Classic Capture Mode

When in classic capture mode for an Oracle Database, Extract supports the
log-retention feature, whereby the database retains the logs that Extract
needs. To enable the log-retention feature, DBLOGIN must be issued with special
privileges before using REGISTER EXTRACT with the LOGRETENTION option. For
simplicity, you can log in as the Extract database user if the correct
privileges were granted to that user when Oracle GoldenGate was installed.
Otherwise, log in as a user with the privileges shown in the following table:
```

```
Oracle Privileges for Log Retention
Oracle EE             How to Grant Privileges version
11.1 and 11.2.0.1     1. Run package to grant Oracle GoldenGate admin privilege.
                    exec dbms_streams_auth.grant_admin_privilege('user')
                    2. Grant the 'become user' privilege.
                    grant become user to user;
11.2.0.2 and later     Run package to grant Oracle GoldenGate admin privilege.
                    exec dbms_goldengate_auth.grant_admin_privilege('user')

Syntax

DBLOGIN {
[SOURCEDB data_source] |
 database@host:port] |
USERID {/ | userid} PASSWORD password]

 [algorithm ENCRYPTKEY {keyname | DEFAULT}] | USERIDALIAS alias [DOMAIN domain]
 | [SYSDBA | SQLID sqlid] [SESSIONCHARSET character_set] }

SOURCEDB data_source

SOURCEDB specifies a data source name. This option is required to identify
one of the following:

* The source or target login database for MySQL and databases that use ODBC.

database@host:port

(MySQL) Specifies a connection string that contains the database name,
host name, and database port number. Can be used to specify a port other
than the default that is specified in the database configuration.

USERID

Supplies a database login credential, if required. Can be used if an Oracle
GoldenGate credential store is not in use. (See the USERIDALIAS option.)
Input varies, depending on the database, as follows:

    userid

    Specifies the name of a database user or a schema, depending on the
    database configuration. For Oracle, a SQL*Net connect string can be
    used. To log into a pluggable database in an Oracle multitenant container
    database, specify userid as a connect string, such as OGGUSER@FINANCE.
    To log into the root container, specify userid as a common user, including
    the C## prefix, such as C##GGADMIN@FINANCE.

    /
    (Oracle) Directs Oracle GoldenGate to use an operating-system login for
    Oracle, not a database user login. Use this argument only if the database
    allows authentication at the operating-system level. To use this option,
    the correct user name must exist in the database, in relation to the
    value of the Oracle OS_AUTHENT_PREFIX initialization parameter. For more
    information, see the USERID | NOUSERID parameter.

    PASSWORD password

    Use when authentication is required to specify the password for the
    database user. If the password was encrypted by means of the ENCRYPT
    PASSWORD command, supply the encrypted password; otherwise, supply the
```

clear-text password. If the password is case-sensitive, type it that way.

If the PASSWORD clause is omitted, you are prompted for a password, and the password is not echoed.

algorithm

If the password was encrypted with the ENCRYPT PASSWORD command, specify the encryption algorithm that was used:

AES128

AES192

AES256

BLOWFISH

ENCRYPTKEY {keyname | DEFAULT}

Specifies the encryption key that was specified with the ENCRYPT PASSWORD command. Use one of the following:

ENCRYPTKEY keyname
Specifies the logical name of a user-created encryption key in the ENCKEYS lookup file. Use if ENCRYPT PASSWORD was used with the KEYNAME option.

ENCRYPTKEY DEFAULT
Directs Oracle GoldenGate to generate a Blowfish key. Use if the ENCRYPT was used with the KEYNAME DEFAULT option.

USERIDALIAS alias [DOMAIN domain]

Supplies a database login credential, if required. Can be used instead of the USERID option if there is a local Oracle GoldenGate credential store that contains a credential with the required privileges for this DBLOGIN command. For more information about using a credential store, see Administering Oracle GoldenGate.

alias

Specifies the alias of a database user credential that is stored in the Oracle GoldenGate credential store. To log into a pluggable database in an Oracle multitenant container database, the user must be stored as a connect string, such as OGGUSER@FINANCE. To log into the root container, the user must be stored as a common user, including the C## prefix, such as C##GGADMIN@FINANCE. For more information about configuring Oracle GoldenGate for a CDB, see the Oracle GoldenGate documentation for your database.

DOMAIN domain

Specifies the credential store domain for the specified alias. A valid domain
entry must exist in the credential store for the specified alias.

SYSDBA

(Oracle) Specifies that the user logs in as sysdba. This option can be used for USERID and USERIDALIAS.

SQLID sqlid

```
(DB2 on z/OS) Issues the SQL command SET CURRENT SQLID = 'sqlid'after the USERID
login (with PASSWORD, if applicable) is completed. If the SET command fails, the
entire DBLOGIN command fails as a unit.

SESSIONCHARSET character_set

(Teradata and MySQL) Sets a database session character set for the
GGSCI connection to the database. All subsequent commands will use the
specified session character set. This command option overrides any SESSIONCHARSET
that is specified in the GLOBALS file.

Examples

Example 1 (Oracle)

DBLOGIN USERIDALIAS alias1

Example 2 (Oracle with non-default domain)

DBLOGIN USERIDALIAS alias1 DOMAIN domain1

Example 3 (Oracle with SYSDBA)

DBLOGIN USERID ogguser@pdb1 SYSDBA password
AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC AES128, ENCRYPTKEY securekey1

Example 4 (MySQL)

DBLOGIN SOURCEDB mysqldb@host1:3305, USERIDALIAS alias1

Example 5 (MySQL)

DBLOGIN SOURCEDB database USERIDALIAS alias1
```

```
---------------------------------------------------------------------
ENCRYPT PASSWORD

Use ENCRYPT PASSWORD to encrypt a password that is used in an Oracle
GoldenGate parameter file or command.

Syntax

ENCRYPT PASSWORD password
[AES128 | AES192 | AES256 | BLOWFISH]
ENCRYPTKEY {key_name | DEFAULT}

password
The login password. Do not enclose the password within quotes. If the
password is case-sensitive, type it that way.

AES128 | AES192 | AES256 | BLOWFISH
Specifies the encryption algorithm to use.

* AES128uses the AES-128 cipher, which has a key size of 128 bits.
```

* AES192uses the AES-192 cipher, which has a key size of 192 bits.

* AES256uses the AES-256 cipher, which has a key size of 256 bits.

* BLOWFISH uses Blowfish encryption with a 64-bit block size and a variable-length key size from 32 bits to 128 bits. Use BLOWFISH only for backward compatibility with earlier Oracle GoldenGate versions.

If no algorithm is specified, AES128 is the default for all database types except DB2 on z/OS, where BLOWFISH is the default. AES is not supported for those platforms.

All of the AES ciphers have a 128-bit block size.

To use AES encryption for any database other than Oracle, the path of the lib sub-directory of the Oracle GoldenGate installation directory must be specified as an environment variable before starting any processes:

* Linux: Specify the path as an entry to the LD_LIBRARY_PATH or SHLIB_PATH variable. For example:

setenv LD_LIBRARY_PATH ./lib:$LD_LIBRARY_PATH

* For Solaris: Specify the path as an entry to the SHLIB_PATH variable.
* For IBMi and AIX: Specify the path as an entry to the LIBPATH variable.
* Windows: Add the path to the PATH variable.

You can use the SETENV parameter to set it as a session variable for the process.

ENCRYPTKEY {key_name | DEFAULT}

Specifies the encryption key.

key_name

Specifies the logical name of a user-created encryption key in a local ENCKEYS lookup file. The key name is used to look up the actual key in the ENCKEYS file. A user-created key and an associated ENCKEYS file is required when using AES encryption; optional, but recommended, for Blowfish encryption.
To use key_name, generate the key with KEYGEN or another utility, then store it in an ENCKEYS file on the source and target systems.

DEFAULT

Directs Oracle GoldenGate to generate a random key that is stored in the trail so that decryption can be performed by the downstream process. This type of key is insecure and should not be used in a production environment. Use this option only when BLOWFISH is specified. ENCRYPT PASSWORD returns an error if DEFAULT is used with any AES algorithm.

Examples

Example 1

ENCRYPT PASSWORD ny14072 BLOWFISH ENCRYPTKEY DEFAULT

Example 2

```
ENCRYPT PASSWORD ny14072 BLOWFISH ENCRYPTKEY superkey3

Example 3

ENCRYPT PASSWORD ny14072 AES192 ENCRYPTKEY superkey2

------------------------------------------------------------------
####################################
#
#DDL COMMANDS
#
#
####################################
------------------------------------------------------------------

DUMPDDL

Use the DUMPDDL command to view the data in the Oracle GoldenGate DDL history
table if the trigger-based DDL capture is in use. This information is the
same information that is used by the Extract process. It is stored in
proprietary
format, but can be exported in human-readable form to the screen or to a
series of SQL tables that can be queried by using regular SQL.

DUMPDDL always dumps all of the records in the DDL history table. Use SQL
queries or search redirected standard output to view information about
particular objects and the operations you are interested in. Because the
history table contains large amounts of data, only the first 4000 bytes
(approximately) of a DDL statement are displayed in order to maintain
efficient performance. The format of the metadata is string based. It is
fully escaped and supports table and column names in their native character
set.

Because the information is historical data that is provided by the DDL before
trigger, it reflects the state of an object before a DDL change. Consequently,
there will not be any data for CREATE operations.

    Note:
    The default name of the before trigger is GGS_DDL_TRIGGER_BEFORE.

Before using DUMPDDL, log into the database as the owner of the history
table by using the DBLOGIN command.

The basic DUMPDDL command outputs metadata to the following tables:

GGS_DDL_OBJECTS         Information about the objects for which DDL operations
                        are being synchronized. SEQNO is the primary key. All
                        of the other tables listed here contain a SEQNO column
                        that is the foreign key to GGS_DDL_OBJECTS.
GGS_DDL_COLUMNS         Information about the columns of the objects involved
                        in DDL synchronization.
GGS_DDL_LOG_GROUPS      Information about the supplemental log groups involved
                        in DDL synchronization.
GGS_DDL_PARTITIONS      Information about the partitions for objects involved
                        in DDL synchronization.
GGS_DDL_PRIMARY_KEYS    Information about the primary keys of the objects
                        involved in DDL synchronization.

The SEQNO column is the DDL sequence number that is listed in the Extract and
Replicat report files. It also can be obtained by querying the DDL history
```

table (default name is GGS_DDL_HIST).

All of these tables are owned by the schema that was designated as the Oracle GoldenGate DDL schema during the installation of the DDL objects. To view the structure of these tables, use the DESC command in SQL*Plus.

Syntax

DUMPDDL [SHOW]

SHOW

Dumps the information contained in the history table to the screen in standard output format. No output tables are produced. All records in the DDL history table are shown.

FLUSH SEQUENCE

Use FLUSH SEQUENCE immediately after you start Extract for the first time during an initial synchronization or a re-synchronization. This command updates an Oracle
sequence so that initial redo records are available at the time that Extract starts to capture transaction data. Normally, redo is not generated until the current cache is exhausted. The flush gives Replicat an initial start point with which to synchronize to the correct sequence value on the target system. From then on, Extract can use the redo that is associated with the usual cache reservation of sequence values.

To Use FLUSH SEQUENCE

The following Oracle Database procedures are used by FLUSH SEQUENCE:

Database Procedure User and Privileges e
Source updateSequence Grants EXECUTE to the owner of the Oracle GoldenGate
DDL objects, or other selected user if not using DDL support.
Target replicateSequen Grants EXECUTE to the Oracle GoldenGate Replicat ce
user.

The sequence.sqlscript installs these procedures. Normally, this script is run as part of the Oracle GoldenGate installation process, but make certain that was done before using FLUSH SEQUENCE. If sequence.sqlwas not run, the flush fails and
an error message similar to the following is generated:
Cannot flush sequence {0}. Refer to the Oracle GoldenGate for Oracle documentation
for instructions on how to set up and run the sequence.sql script. Error {1}.

2. The GLOBALS file must contain a GGSCHEMA parameter that specifies the schema in which the procedures are installed. This user must have CONNECT, RESOURCE, and DBA
privileges.

3. Before using FLUSH SEQUENCE, issue the DBLOGIN command as the database user that has EXECUTE privilege on the updateSequence procedure. If logging into a multitenant container database, log into the pluggable database that contains the sequence that is to be flushed.

Note:
For full instructions on configuring Oracle GoldenGate to support sequences, see the Oracle GoldenGate documentation for your databaseDatabase.

Syntax

```
FLUSH SEQUENCE owner.sequence
```

owner.sequence

The owner and name of an Oracle sequence. The schema name cannot be null. You can use an asterisk (*) wildcard for the sequence name but not for the owner name.

Example

```
FLUSH SEQUENCE scott.seq*
```

LIST TABLES

Use LIST TABLES to list all tables in the database that match the specification provided with the command argument. Use the DBLOGIN command to establish a database connection before using this command. If logging into an Oracle multitenant container database, log in to the pluggable database that contains the tables that you want to list.

Syntax

```
LIST TABLES table
```

table

The name of a table or a group of tables specified with a wildcard (*).

Example

The following shows a LIST TABLES command and sample output.

```
list tables tcust*

TCUSTMER
TCUSTORD
```

MININGDBLOGIN

Use MININGDBLOGIN to establish a connection to a downstream Oracle Database logmining server in preparation to issue other Oracle GoldenGate commands that affect this database, such as REGISTER EXTRACT. Use this command only if establishing Extract in integrated capture mode for an Oracle Database.

To log into a source Oracle Database that serves as the database logmining server, use the DBLOGIN command. MININGDBLOGIN is reserved for login to a downstream mining database.

The user who issues MININGDBLOGIN must:

* Have privileges granted through the Oracle dbms_goldengate_auth.grant_admin_privilege procedure.

```
* Be the user that is specified with the TRANLOGOPTIONS MININGUSER parameter
for the Extract group that is associated with this MININGDBLOGIN.

* Not be changed while Extract is in integrated capture mode.


For support and configuration information for integrated capture, see the
Oracle GoldenGate documentation for your database.

Syntax

MININGDBLOGIN {

USERID {/ | userid} PASSWORD password] [algorithm ENCRYPTKEY {keyname |
DEFAULT}] | USERIDALIAS alias [DOMAIN domain] |
[SYSDBA]
}
USERID
Supplies a database login credential. Can be used if an Oracle GoldenGate
credential store is not in use. (See the USERIDALIAS option.) Input varies,
depending on the database, as follows:

    userid

    Specifies the name of a database user or a SQL*Net connect string. To log
    into a pluggable database in an Oracle multitenant container database,
    specify useridas a connect string, such as OGGUSER@FINANCE. To log into
    the root container, specify useridas a common user, including the C##
    prefix, such as C##GGADMIN@FINANCE. For more information about configuring
    Oracle GoldenGate for a CDB, see the Oracle GoldenGate documentation for
    your database.

    /
    (Oracle) Directs Oracle GoldenGate to use an operating-system login for
    Oracle, not a database user login. Use this argument only if the database
    allows authentication at the operating-system level. To use this option,
    the correct user name must exist in the database, in relation to the value
of
    the Oracle OS_AUTHENT_PREFIX initialization parameter. For more information,
    see the USERID | NOUSERID parameter.

    PASSWORD password

    Use when authentication is required to specify the password for the database
    user. If the password was encrypted by means of the ENCRYPT PASSWORD
command,
    supply the encrypted password; otherwise, supply the clear-text password. If
    the password is case-sensitive, type it that way.

    If the PASSWORD clause is omitted, you are prompted for a password, and the
    password is not echoed.

    algorithm

    If the password was encrypted with the ENCRYPT PASSWORD command, specify the
    encryption algorithm that was used:

    AES128
    AES192
    AES256
    BLOWFISH
```

```
    ENCRYPTKEY {keyname | DEFAULT}

    Specifies the encryption key that was specified with the ENCRYPT PASSWORD
    command. Use one of the following:

        ENCRYPTKEY keyname

        Specifies the logical name of a user-created encryption key in the
ENCKEYS
        lookup file. Use if ENCRYPT PASSWORD was used with the KEYNAME option.

        ENCRYPTKEY DEFAULT

        Directs Oracle GoldenGate to generate a Blowfish key. Use if the ENCRYPT
        PASSWORD command was used with the KEYNAME DEFAULT option.

USERIDALIAS alias [DOMAIN domain]

Supplies the alias of a database login credential. Can be used instead of the
USERID option if there is a local Oracle GoldenGate credential store that
contains a credential with the required privileges for this MININGDBLOGIN
command. For more information about using a credential store, see Administering
Oracle GoldenGate.

To log into a pluggable database in an Oracle multitenant container database,
the user must be stored as a connect string, such as OGGUSER@FINANCE. To log
into the root container, the user must be stored as a common user, including
the C## prefix, such as C##GGADMIN@FINANCE. For more information about
configuring Oracle GoldenGate for a CDB, see the Oracle GoldenGate documentation
for your database.

    alias

    Specifies the alias of a database user credential that is stored in the
    Oracle GoldenGate credential store. The user that is specified with
    USERIDALIAS must be the common database user.

    DOMAIN domain

    Specifies the credential store domain for the specified alias. A valid
    domain entry must exist in the credential store for the specified alias.

SYSDBA

(Oracle) Specifies that the user logs in as sysdba. This option can be used
for USERID and USERIDALIAS.

Examples

Example 1

MININGDBLOGIN USERIDALIAS oggalias SESSIONCHARSET ISO-8859-11

Example 2

MININGDBLOGIN USERID ogg@ora1.ora, PASSWORD
AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC AES128, ENCRYPTKEY securekey1


------------------------------------------------------------------------
```

```
SET NAMECCSID

Use NAMECCSID to set the CCSID (coded character set identifier) of the
GGSCI session when you need to issue commands for tables in a DB2 for i
database. This command is required if the CCSID of the object names stored
in the SQL catalog tables is different from the CCSID of the system. The SQL
catalog tables are created with the CCSID of the system, but the actual
database object names could be represented with a different CCSID. The
catalog does not indicate this difference when queried, and therefore
Oracle GoldenGate could retrieve the name incorrectly unless NAMECCSIDis
present to supply the correct CCSID value.

To set the CCSID for GLOBALS, Extract, Replicat, or DEFGEN, use the
NAMECCSID parameter.

SET NAMECCSID is not valid if the DBLOGIN command was previously issued,
because that command affects the GGSCI session. To issue SET NAMECCSID
after a DBLOGIN command, restart GGSCI.

To view the current CCSID, use the SHOW command. If the CCSID is not set
through the GGSCI session or through the parameter NAMECCSID, the SHOW
value will be DEFAULT.

Syntax

SET NAMECCSID {CCSID | DEFAULT}

CCSID

A valid DB2 for i coded character set identifier that is to be used for
the GGSCI session.

DEFAULT

Indicates that the system CCSID is to be used for the GGSCI session.

Example

SET NAMECCSID 1141

------------------------------------------------------------------------
####################################
#
#TRANDATA COMMANDS
#
#
####################################

------------------------------------------------------------------------
ADD SCHEMATRANDATA

Valid for Oracle. Use ADD SCHEMATRANDATA to enable schema-level
supplemental logging for a schema. ADD SCHEMATRANDATA acts on all of the
current and future tables in a given schema to automatically log a superset
of available keys that Oracle GoldenGate needs for row identification.

ADD SCHEMATRANDATAis valid for both integrated and classic capture and
does the following:

* Enables Oracle supplemental logging for new tables created with a
CREATE TABLE.
```

* Updates supplemental logging for tables affected by an ALTER TABLE
to add or drop columns.

* Updates supplemental logging for tables that are renamed.

* Updates supplemental logging for tables for which unique or primary keys
are added or dropped.

By default, ADD SCHEMATRANDATA enables the unconditional logging of the
primary key and the conditional supplemental logging of all unique keys
and foreign keys of any table within the specified schema in the following
order:

1. Primary key.

2. Even if the primary key exists, the unique indexes are supplementally logged.
This is, because the default for Oracle are the scheduling columns!. Unique keys
that
contain ADT member columns are also logged. Only unique keys on virtual
columns (function-based indexes) are not logged.

3. If none of the preceding exists, all scalar columns of the table are
logged. (System-generated row-OIDs are always logged.)

ADD SCHEMATRANDATA also supports the conditional or unconditional logging
requirements for using integrated Replicat.

    Note:
    Apply Oracle Patch 10423000 to the source database if the Oracle version
    is earlier than 11.2.0.2.

When to Use ADD SCHEMATRANDATA
ADD SCHEMATRANDATA must be used in the following cases:

* For all tables that are part of an Extract group that is to be
configured for integrated capture. ADD SCHEMATRANDATA ensures that the
correct key is logged by logging all of the keys.

* For all source tables that will be processed in an integrated Replicat
group. Options are provided that enable the logging of the primary, unique,
and foreign keys to support the computation of dependencies among relational
tables being processed through different apply servers.

* When DDL replication is active and DML is concurrent with DDL that creates
new tables or alters key columns. It best handles scenarios where DML can be
applied to objects very shortly after DDL is issued on them. ADD SCHEMATRANDATA
causes the appropriate key values to be logged in the redo log atomically with
each DDL operation, thus ensuring metadata continuity for the DML when it is
captured from the log, despite any lag in Extract processing.

Database-level Logging Requirements for Using ADD SCHEMATRANDATA

Supplemental Logging is only managed (controlled) on the source database.
Minimal supplemental Logguing is a must for the source database so you must put
the source database into forced logging mode and enable minimal supplemental
logging at the database level when using Oracle GoldenGate. This adds row
chaining information, if any exists, to the redo log for update operations.

Additional Considerations for Using ADD SCHEMATRANDATA

* Before using ADD SCHEMATRANDATA, issue the DBLOGIN command. The user who issues the command must be granted an Oracle GoldenGate administrator user.

SQL> exec dbms_goldengate_auth.grant_admin_privilege('user')

* ADD SCHEMATRANDATA can be used instead of the ADD TRANDATA command when DDL replication is not enabled. Note, however, that if a table has no primary key but has multiple unique keys, ADD SCHEMATRANDATA causes the database to log all of the unique keys. In such cases, ADD SCHEMATRANDATA causes the database to log more redo data than does ADD TRANDATA. To avoid the extra logging, designate one of the unique keys as a primary key, if possible.

* For tables with a primary key, with a single unique key, or without a key, ADD SCHEMATRANDATA adds no additional logging overhead, as compared to ADD TRANDATA.

For more information, see ADD TRANDATA.

* If you must log additional, non-key columns of a specific table (or tables) for
use by Oracle GoldenGate, such as those needed for FILTER statements and KEYCOLS clauses in the TABLE and MAP parameters, issue an ADD TRANDATA command for those columns. That command has a COLS option to issue table-level supplemental logging
for the columns, and it can be used in conjunction with ADD SCHEMATRANDATA.

Syntax

ADD SCHEMATRANDATA schema {
[ALLOWNONVALIDATEDKEYS]
[NOSCHEDULINGCOLS | ALLCOLS]}
[NOVALIDATE]
[PREPARECSN {WAIT | LOCK | NOWAIT | NONE}]

schema

The schema for which you want the supplementary key information to be logged. Do not use a wildcard. To issue ADD SCHEMATRANDATA for schemas in more than one pluggable database of a multitenant container database, log in to each pluggable database separately with DBLOGINand then issue ADD SCHEMATRANDATA.

ALLOWNONVALIDATEDKEYS

This option is valid for Oracle 11.2.0.4 and later 11g versions and Oracle 12.1.0.2
and later 12c versions. (Not valid for Oracle 11.2.0.3 or 12.1.0.1.) It includes NON
VALIDATED and NOT VALID primary keys in the supplemental logging. These keys override
the normal key selection criteria that is used by Oracle GoldenGate. If the GLOBALS
parameter ALLOWNONVALIDATEDKEYS is being used, ADD SCHEMATRANDATA runs with ALLOWNONVALIDATEDKEYS whether or not it is specified. By default, NON VALIDATED and NOT VALID primary keys are not logged.

NOSCHEDULINGCOLS | ALLCOLS

```
You can use these options together though the latter option is used. For
example, with the
ADD SCHEMATRANDATA oggadm_ext ALLCOL NOSCHEDULINGCOLS command the
NOSCHEDULINGCOLS
option would be used. By default, ADD SCHEMATRANDATA enables:

    * unconditional logging of the primary key
    * conditional supplemental logging of all unique key(s)
    * conditional supplemental logging of all foreign key(s) of all current and
      future tables in the given schema.

Unconditional log groups log the before images of the specified columns when the
table is
changed, regardless of whether the change affected any of the specified columns.
Unconditional
log groups are sometimes referred to as "always log groups." Conditional log
groups log the
before images of all specified columns only if at least one of the columns in
the log group
is changed.

    NOSCHEDULINGCOLS

    Disables the logging of scheduling columns. Unconditional logging forces the
primary
    key values to the log whether or not the key was changed in the current
    operation. Conditional logging logs all of the column values of a foreign or
    unique key if at least one of them was changed in the current operation. The
    integrated Replicat primary key, unique keys, and foreign keys must all be
    available to the inbound server to compute dependencies. Setting
supplemental
    logging on the source database impacts the target database when identifying
the
    unique row and dependency calculations (based on the key columns at the
target
    system).

    ALLCOLS

    Enables the unconditional supplemental logging of all supported key and non-
key
    columns for all current and future tables in the given schema. This option
    enables the logging of the keys required to compute dependencies, plus
columns
    that are required for filtering, conflict resolution, or other purposes.
Columns
    like LOB, LONG, and ADT are not included.

NOVALIDATE

Valid for all databases supported by ADD SCHEMATRANDATA.
Suppresses additional information about the table being handled being processed
by ADD SCHEMATRANDATA. By default, this option is enabled. The additional
information processing creates a lapse time on command response so this
option can be used to increase response time.

PREPARECSN {WAIT | LOCK | NOWAIT | NONE}

Valid for Oracle for both DML and DDL. Automatically prepares the tables at the
source
so the Oracle data pump export dump file will includes Instantiation CSNs.
```

```
Replicat uses
the per table instantiation CSN set by the Oracle Datapump (on import) to filter
out
trail records. On the target, the data pump import populates the system tables
and
views with instantiation SCNs using the DBOPTIONS ENABLE_INSTANTIATION_FILTERING
parameter to enable table-level instantiation filtering.

    WAIT

    Wait for any in-flight transactions and prepare table instantiation.

    LOCK

    Put a lock on the table (to prepare for table instantiation).

    NOWAIT

    Default behavior, preparing for instantiation is done immediately.

    NONE

    No instantiation preparation occurs.

Example

Example 1

The following enables supplemental logging for the schema scott.

ADD SCHEMATRANDATA scott

Example 2

The following example logs all supported key and non-key columns for all current
and future tables in the schema named scott.

ADD SCHEMATRANDATA scott ALLCOLS
Example 3


The following example suppress additional table information processing.

ADD SCHEMATRANDATA acct.emp* NOVALIDATE


------------------------------------------------------------------------
ADD TRANDATA

Use ADD TRANDATA to enable Oracle GoldenGate to acquire the transaction
information
that it needs from the transaction records.

Before using this command, use the DBLOGIN command to establish a database
connection.

ADD TRANDATA is valid only for the source databases that are listed here. For
other
supported databases, this functionality may exist already or must be configured
through the database interface. See the Oracle GoldenGate configuration guide
for
```

your database for any special requirements that apply to making transaction information available.

DB2 for i Databases

Use ADD TRANDATA to start the journaling of data. The ADD TRANDATA command calls STRJRNP F and is the recommended method to start journaling for tables, because it
ensures that the required journal image attribute of Record Images (IMAGES): *BOTH
is set on the STRJRNPF command.

DB2 LUW Database

Use ADD TRANDATA to enable DATA CAPTURE CHANGESon specified tables. By default, ADD
TRANDATA issues the following command to the database:
ALTER TABLE name DATA CAPTURE CHANGES INCLUDE LONGVAR COLUMNS;
You can exclude the LONGVAR clause by using ADD TRANDATA with the EXCLUDELONG option.

DB2 z/OS Database

Use ADD TRANDATA to enable DATA CAPTURE CHANGESon specified tables. By default, ADD
TRANDATA issues the following command to the database:
ALTER TABLE name DATA CAPTURE CHANGES;

Oracle Database

By default, ADD TRANDATA for Oracle enables the unconditional logging of the primary
key and the conditional supplemental logging of all unique key(s) and foreign key(s)
of the specified table.

If possible, use the ADD SCHEMATRANDATA command rather than the ADD TRANDATA command.
The ADD SCHEMATRANDATA command ensures replication continuity should DML occur on an
object for which DDL has just been performed. You can exclude objects from the schema specification by using the exclusion parameters.

To use the Oracle GoldenGate DDL replication feature, you must use the ADD SCHEMATRANDATA command to log the required supplemental data.

When using ADD SCHEMATRANDATA, you can use ADD TRANDATA with the COLS option to any non-key columns, such as those needed for FILTER statements and KEYCOLS clauses
in the TABLE and MAP parameters.

    Note:
    It is possible to use ADD TRANDATA for Oracle when DDL support is enabled, but
    only if you can stop DML on all tables before DDL is performed on them or, if
    that is not possible, you can guarantee that no users or applications will issue
    DDL that adds new tables whose names satisfy an object specification in a TABLE
    or MAP statement. There must be no possibility that users or applications

```
will
    issue DDL that changes the key definitions of any tables that are already in
    the Oracle GoldenGate configuration.

Supplemental Logging is only managed (controlled) on the source database.
Minimal supplemental Logguing is a must for the source database so you must put
the source database into forced logging mode and enable minimal supplemental
logging at the database level when using Oracle GoldenGate. This adds row
chaining information, if any exists, to the redo log for update operations.

Take the following into account when using ADD TRANDATA for an Oracle Database:

* If any of the logging details change after Oracle GoldenGate starts extracting
data, you must stop and then start the Extract process that is reading from the
affected table before any data is changed.

* When enabling supplemental logging with ADD TRANDATA, Oracle GoldenGate
creates
at least a supplemental log group with the extention of the object ID to a
prefix of
GGS_. For example GGS_18342. Depending on the configuration options, additional
supplemental log groups may appear.



Syntax

ADD TRANDATA {[container.]owner.table | schema.table [JOURNAL
library/journal] | library/file [JOURNAL library/journal]}
 [NOSCHEDULINGCOLS | ALLCOLS]
 [COLS (columns)]
 [INCLUDELONG | EXCLUDELONG]
 [LOBSNEVER | LOBSALWAYS | LOBSIFCHANGED | LOBSALWAYSNOINDEX]
 [NOKEY]
 [NOVALIDATE]
 [PREPARECSN  {WAIT | LOCK | NOWAIT | NONE}]

[container.]owner.table

Valid for DB2 LUW, DB2 for z/OS, Oracle.

The two-part or three-part name specification. Use a two-part name of
owner.table
for all supported databases except an Oracle multitenant container database. Use
a
three-part name of container.owner.table for an Oracle multitenant container
database.
A wildcard can be used for any component. Used with a wildcard, ADD TRANDATA
filters
out names that match the names of system objects. To use ADD TRANDATA for
objects that
are not system objects but have names that match those of system objects in a
wildcard
pattern, issue ADD TRANDATA for those objects without using a wildcard.

schema.table [JOURNAL library/journal] |

library/file [JOURNAL library/journal]

Valid for DB2 for i.
```

```
Specifies the SQL schema and name of a table or the native library and file
name.
If a default journal is set with the DEFAULT JOURNAL command, you can omit the
JOURNAL
option; otherwise it is required.

NOSCHEDULINGCOLS | ALLCOLS

Valid for Oracle
These options satisfy the logging requirements of an integrated Replicat that
will
be processing the tables that you are specifying with ADD TRANDATA. By default,
ADD
TRANDATA enables:

    * unconditional logging of the primary key
    * conditional supplemental logging of all unique key(s)
    * conditional supplemental logging of all foreign key(s) of all current and
      future tables in the given schema.

Unconditional log groups log the before images of the specified columns when the
table is
changed, regardless of whether the change affected any of the specified columns.
Unconditional
log groups are sometimes referred to as "always log groups." Conditional log
groups log the
before images of all specified columns only if at least one of the columns in
the log group
is changed.

    NOSCHEDULINGCOLS

    Disables the logging of scheduling columns. The primary key, unique keys,
and foreign
    keys are needed for the dependency computation of the integrated Replicat

    ALLCOLS

    Enables the unconditional supplemental logging of all of the key and non-key
    columns of the table. This option enables the logging of the keys required
to
    compute dependencies, plus all other columns for use in filtering, conflict
    resolution, or other purposes.

COLS (columns)

Valid for Oracle Database.
Use the COLS option to log specific non-key columns. Can be used to log columns
specified in a KEYCOLS clause and to log columns that will be needed for
filtering
or manipulation purposes, which might be more efficient than fetching those
values
with a FETCHCOLS clause in a TABLE statement. Separate multiple columns with
commas,
for example NAME, ID, DOB.

INCLUDELONG | EXCLUDELONG

Valid for DB2 LUW.

Controls whether or not the ALTER TABLE issued by ADD TRANDATA includes the
```

```
INCLUDE
LONGVAR COLUMNS attribute. INCLUDELONG is the default. When ADD TRANDATA is
issued with
this option, Oracle GoldenGate issues the following statement:
ALTER TABLE name DATA CAPTURE CHANGES INCLUDE LONGVAR COLUMNS;
When EXCLUDELONG is used, the following is the command:

ALTER TABLE name DATA CAPTURE CHANGES;

When EXCLUDELONG is used, Oracle GoldenGate does not support functionality that
requires
before images of tables that include LONGVAR columns. For example, the
GETUPDATEBEFORES
parameter. To support this functionality, changes to LONGVAR columns in the
transaction
logs must include both the before and after images of the column value.

LOBSNEVER | LOBSALWAYS | LOBSIFCHANGED | LOBSALWAYSNOINDEX

Note:
The ADD TRANDATA command will overwrite the LOB setting that is currently set
for
the table. To change the setting afterwards, you must use the sp_setrepcol
script.

    LOBSNEVER

    Prevents LOB data from being propagated.
    Note this exception: If the LOB column
    is inserted with a NULL value, or if it is skipped in an INSERT operation,
then
    Extract will write that column to the trail with NULL data.

    LOBSALWAYS

    Does two things: it uses sp_setrepcol to set LOB replication to
ALWAYS_REPLICATE
    (always replicate LOB data whether or not it has changed in a transaction),
and
    it marks the table to use an index on replication (by means of the USE_INDEX
    of sp_setreptable). Because a LOB is marked for replication in a single
transaction,
    this can take a long time, and USE_INDEX reduces that time by creating a
global
    nonclustered index for every LOB. A shared-table lock is held while the
global
    nonclustered index is created.

    LOBSIFCHANGED

    Replicates LOB data only if it was changed during a transaction. This
reduces
    replication overhead but does not protect against inconsistencies that could
    occur on the target outside the replication environment. This is the
default.

    LOBSALWAYSNOINDEX

    Sets LOB replication to ALWAYS_REPLICATE (always replicate LOB data whether
or not
    it has changed in a transaction). This adds overhead, but protects against
```

```
    inconsistencies that could occur on the target outside the replication
environment.
    LOBSALWAYSNOINDEX does not mark the table to use an index on replication.
The
    benefit is that no lock is held while ADD TRANDATA is being executed.

Note:

When using the ALWAYS_REPLICATE option, if a LOB column contains a NULL value,
and
then another column in the table gets updated (but not the LOB), that LOB will
not
be captured even though ALWAYS_REPLICATE is enabled.
You can check the LOB settings of a table with the INFO TRANDATA command, after
ADD
TRANDATA has been used for that table. It shows the LOB settings for all of the
LOB
columns.

NOKEY

Valid for all databases supported by ADD TRANDATA.

Suppresses the supplemental logging of primary key columns. If using NOKEY, use
the
COLS option to log alternate columns that can serve as keys, and designate those
columns as substitute keys by using the KEYCOLS option of the TABLE or MAP
parameter.

NOVALIDATE

Valid for all databases supported by ADD TRANDATA.
Suppresses additional information about the table being handled being processed
by ADD TRANDATA. By default, this option is enabled. The additional information
processing creates a lapse time on command response so this option can be used
to
increase response time.

PREPARECSN {WAIT | LOCK | NOWAIT | NONE}

Valid for Oracle for both DML and DDL. Automatically prepares the tables at the
source
so the Oracle Datapump Export dump file will includes Instantiation CSNs.
Replicat
uses the per table instantiation CSN set by the Oracle Datapump (on import) to
filter
out trail records. On the target, the data pump import populates the system
tables
and views with instantiation SCNs using the DBOPTIONS
ENABLE_INSTANTIATION_FILTERING
parameter to enable table-level instantiation filtering.

    WAIT

    Wait for any in-flight transactions and prepare table instantiation.

    LOCK

    Put a lock on the table (to prepare for table instantiation).

    NOWAIT
```

```
    Default behavior, preparing for instantiation is done immediately.

    NONE

    No instantiation preparation occurs.

Examples

Example 1

The following example causes the primary key to be logged for an Oracle table.

ADD TRANDATA finance.acct

Example 2

The following example enables the unconditional supplemental logging of all of
the
key and non-key columns for the table named acct.

ADD TRANDATA acct ALLCOLS

Example 3

The following Oracle example causes the primary key to be logged plus the non-
key
columns name and address.

ADD TRANDATA finance.acct, COLS (name, address)

Example 4

The following Oracle example prevents the primary key from being logged, but
logs
the non-key columns name and pid instead.

ADD TRANDATA finance.acct, NOKEY, COLS (name, pid)

Example 5

The following example adds logging though does not prepare the table for
instantiation.

ADD TRANDATA acct PREPARECSN NONE

Example 6

The following example suppress additional table information processing.

ADD TRANDATA acct.emp* NOVALIDATE
------------------------------------------------------------------
DELETE SCHEMATRANDATA

DELETE SCHEMATRANDATA

Use DELETE SCHEMATRANDATA to remove the Oracle schema-level supplemental logging
that was added with the ADD SCHEMATRANDATA command. Use the DBLOGIN command to
establish a database connection before using this command. The user that is
specified with this command must have the privilege to remove supplemental log
groups.
```

By default, this command attempts to remove the supplemental logging of the key columns that are used by Oracle GoldenGate (can be the primary key, a unique key,
KEYCOLS columns, or all columns) and also the scheduling columns. The scheduling columns are the primary key, all of the unique keys, and all of the foreign keys.
To delete the logging of the Oracle GoldenGate key columns, but not the scheduling
columns, include the NOSCHEDULINGCOLS option with DELETE SCHEMATRANDATA. If ADD SCHEMATRANDATA was issued with the ALLCOLS option, use DELETE SCHEMATRANDATA with
the ALLCOLS option to remove the supplemental logging of all of the columns, including the Oracle GoldenGate key columns.

Syntax

```
DELETE SCHEMATRANDATA schema [NOSCHEDULINGCOLS | ALLCOLS]
```

schema

The schema for which you want supplemental logging to be removed. Do not use a wildcard. If the source is an Oracle multitenant container database, make certain
to log into the pluggable database that contains the schema for which you want to remove the logging. See DBLOGIN for more information.

NOSCHEDULINGCOLS

Prevents the command from removing the supplemental logging of the scheduling columns of the tables in the specified schema. The scheduling columns are the primary key, all of the unique keys, and all of the foreign keys of a table.

ALLCOLS

Removes the supplemental logging of all of the columns of the tables in the specified schema.

Examples

Example 1

```
DELETE SCHEMATRANDATA scott
```

Example 2

```
DELETE SCHEMATRANDATA scott ALLCOLS
```

DELETE TRANDATA

DELETE TRANDATA

Use DELETE TRANDATA to do one of the following:

* DB2 LUW and DB2 on z/OS: Alters the table to DATA CAPTURE NONE.

* Oracle: Disable supplemental logging.

By default, this command attempts to remove the supplemental logging of the

key columns that are used by Oracle GoldenGate (can be the primary key, a unique key, KEYCOLS columns, or all columns) and also the scheduling columns. The scheduling columns are the primary key, all of the unique keys, and all of the foreign keys. To delete the logging of the Oracle GoldenGate key columns, but not the scheduling columns, include the NOSCHEDULINGCOLS option with DELETE TRANDATA. If ADD TRANDATA was issued with the ALLCOLS option, use DELETE TRANDATA with the ALLCOLS option to remove the supplemental logging of all of the columns,
including the Oracle GoldenGate key columns.
Use the DBLOGIN command to establish a database connection before using this command. The user specified with this command must have the same privileges that are required for ADD TRANDATA.

Syntax

```
DELETE TRANDATA [container.]owner.table [NOSCHEDULINGCOLS | ALLCOLS]
```

[container.]owner.table

The pluggable database (if this is an Oracle multitenant container database), owner and name of the table or file. A wildcard can be used for any name component.

NOSCHEDULINGCOLS

Prevents the command from removing the supplemental logging of the scheduling columns of the specified table. The scheduling columns are the primary key, all of the unique keys, and all of the foreign keys of a table.

ALLCOLS

Removes the supplemental logging of all of the columns of the specified table.

Examples

Example 1

```
DELETE TRANDATA finance.acct
```

Example 2

```
DELETE TRANDATA finance.ac*
```

Example 3

```
DELETE TRANDATA finance.acct ALLCOLS
```

INFO SCHEMATRANDATA

INFO SCHEMATRANDATA

Use INFO SCHEMATRANDATA to determine whether Oracle schema-level supplemental logging is enabled for the specified schema or if any instantiation information is available. Use the DBLOGIN command to establish a database connection before using this command.

Syntax

```
INFO SCHEMATRANDATA schema
```

schema

The schema for which you want to confirm supplemental logging. Do not use a wildcard.
To get information on the appropriate schema in an Oracle multitenant container database, make certain to log into the correct pluggable database with DBLOGIN.

Example

```
INFO SCHEMATRANDATA scott
------------------------------------------------------------------------
INFO TRANDATA
```

INFO TRANDATA

Use INFO TRANDATA to get the following information:

* DB2 LUW and DB2 on z/OS: Determine whether DATA CAPTURE is enabled or not.

* Oracle: Determine whether supplemental logging is enabled, and to show the names of columns that are being logged supplementally. If all columns are being logged, the notation ALL is displayed instead of individual column names.

Displays any SCN instantiation information.

Use the DBLOGIN command to establish a database connection before using this command.

Syntax

```
INFO TRANDATA [container.]owner.table
```

[container.]owner.table

The pluggable database (if this is an Oracle multitenant container database), owner
and name of the table or file for which you want to view trandata information. The
owner is not required if it is the same as the login name that was specified by the
DBLOGIN command. A wildcard can be used for the table name but not the owner name.

NOVALIDATE

Suppresses additional information about the table being handled being processed by
ADD TRANDATA. By default, this option is enabled. The additional information processing creates a lapse time on command response so this option can be used to increase response time.

Examples

Example 1

```
INFO TRANDATA finance.acct
```

Example 2

```
INFO TRANDATA finance.ac*
```

```
Example 3

INFO TRANDATA finance.ac* NOVALIDATE

####################################
#
#CHECKPOINT TABLE COMMANDS
#
#
####################################
------------------------------------------------------------------------
ADD CHECKPOINTTABLE

Not valid for Replicat for Java, Oracle GoldenGate Applications Adapter, or
Oracle GoldenGate for Big Data.

Use ADD CHECKPOINTTABLE to create a checkpoint table in the target database.
Replicat uses the table to maintain a record of its read position in the trail
for recovery purposes.

The use of a checkpoint table is strongly recommended, because it causes
checkpoints to be part of the Replicat transaction. This allows Replicat to
recover more easily in certain circumstances than when a checkpoint file alone
is used. However, do not use a checkpoint table when configuring Replicat to
operate in integrated mode against an Oracle target database. It is not
required in that mode.

One table can serve as the default checkpoint table for all Replicat groups in
an Oracle GoldenGate instance if you specify it with the CHECKPOINTTABLE
parameter
in a GLOBALS file. More than one instance of Oracle GoldenGate (multiple
installations)
can use the same checkpoint table. Oracle GoldenGate keeps track of the
checkpoints
even when the same Replicat group name exists in different instances.

Use the DBLOGIN command to establish a database connection before using this
command.
Do not change the names or attributes of the columns in this table. You may,
however,
change table storage attributes.

For more information about using a checkpoint table, see Administering Oracle
GoldenGate.

Syntax

ADD CHECKPOINTTABLE [[container. | catalog.]owner.table]

container. | catalog.

The Oracle pluggable database, if applicable. If this option is
omitted, the catalog or pluggable database defaults to the one that is
associated with
the SOURCEDB, USERID, or USERIDALIAS portion of the DBLOGIN command (depending
on
the database).

owner.table
```

The owner and name of the checkpoint table to be created. The name cannot contain any
special characters, such as quotes, backslash, dollar sign, and percent symbol. The name
of a MySQL checkpoint table can contain no more than 30 characters.

The owner and name can be omitted if you are using this table as the default checkpoint
table and it is listed with CHECKPOINTTABLE in the GLOBALS file.

It is recommended, but not required, that the table be created in a schema dedicated to
Oracle GoldenGate. If an owner and name are not specified, a default table is created
based on the CHECKPOINTTABLE parameter in the GLOBALS parameter file.

Record the name of the table, because you will need it to view statistics or delete the
table if needed.

Examples

Example 1

The following adds a checkpoint table with the default name specified in the GLOBALS
file.

```
ADD CHECKPOINTTABLE
```

Example 2

The following adds a checkpoint table with a user-defined name.

```
ADD CHECKPOINTTABLE ggs.fin_check
```

CLEANUP CHECKPOINTTABLE

CLEANUP CHECKPOINTTABLE

Not valid for Replicat for Java, Oracle GoldenGate Applications Adapter, or Oracle
GoldenGate Big Data.

Use CLEANUP CHECKPOINTTABLE to remove checkpoint records from the checkpoint table
when there is no checkpoint file associated with it in the working Oracle GoldenGate
directory (from which GGSCI was started). The purpose of this command is to remove
checkpoint records that are not needed any more, either because groups were changed
or files were moved.

Use the DBLOGIN command to establish a database connection before using this command.

Syntax

```
CLEANUP CHECKPOINTTABLE [[container. | catalog.]owner.table]
```

```
container. | catalog.

The Oracle pluggable database, if applicable. If this option is omitted, the
catalog
or pluggable database defaults to the one that is associated with the SOURCEDB,
USERID,
or USERIDALIAS portion of the DBLOGIN command (depending on the database).

owner.table

The owner and name of the checkpoint table to be cleaned up. If an owner and
name are
not specified, the table that is affected is the one specified with the
CHECKPOINTTABLE
parameter in the GLOBALS parameter file.

Example

CLEANUP CHECKPOINTTABLE ggs.fin_check

------------------------------------------------------------------------
DELETE CHECKPOINTTABLE

DELETE CHECKPOINTTABLE

Not valid for Replicat for Java, Oracle GoldenGate Applications Adapter, or
Oracle
GoldenGate Big Data.

Use DELETE CHECKPOINTTABLE to drop a checkpoint table from the database. Use the
DBLOGIN
command to establish a database connection before using this command.

To stop using a checkpoint table while the associated Replicat group remains
active,
follow these steps:

1. Run GGSCI.

2. Stop Replicat.

   STOP REPLICAT group

3. Delete the Replicat group and then add it back with the following commands.

   DELETE REPLICAT group
   ADD REPLICAT group, EXTTRAIL trail, NODBCHECKPOINT

4. Exit GGSCI, then start it again.

5. Start Replicat again.

   START REPLICAT group

6. Log into the database with the DBLOGIN command, using the appropriate
authentication
options for the database.

7. Delete the checkpoint table with DELETE CHECKPOINTTABLE.
```

If the checkpoint table is deleted while Replicat is still running and transactions
are occurring, Replicat will abend with an error that the checkpoint table could not
be found. However, the checkpoints are still maintained on disk in the checkpoint file.
To resume processing, add the checkpoint table back under the same name. Data in the
trail resumes replicating. Then, you can delete the checkpoint table.

Syntax

```
DELETE CHECKPOINTTABLE [[container. | catalog.]owner.table] [!]
```

container. | catalog.

The Oracle pluggable database, if applicable. If this option is omitted, the catalog
or pluggable database defaults to the one that is associated with the SOURCEDB, USERID,
or USERIDALIAS portion (depending on the database) of the DBLOGIN command.

owner.table

The owner and name of the checkpoint table to be deleted. An owner and name are not
required if they are the same as those specified with the CHECKPOINTTABLE parameter
in the GLOBALS file.

!
Bypasses the prompt that confirms intent to delete the table.

Example

```
DELETE CHECKPOINTTABLE ggs.fin_check
------------------------------------------------------------------
INFO CHECKPOINTTABLE
```

INFO CHECKPOINTTABLE

Not valid for Replicat for Java, Oracle GoldenGate Applications Adapter, or Oracle
GoldenGate Big Data.

Use INFO CHECKPOINTTABLE to confirm the existence of a checkpoint table and view the
date and time that it was created. It returns a message similar to the following:

```
Checkpoint table HR.CHKPT_TBLE created 2017-01-06 11:51:53.
```

Use the DBLOGIN command to establish a database connection before using this command.

Syntax

```
INFO CHECKPOINTTABLE [[container. | catalog.]owner.table]
```

container. | catalog.

```
The Oracle pluggable database, if applicable. If this option is omitted, the
catalog or
pluggable database defaults to the one that is associated with the SOURCEDB,
USERID,
or USERIDALIAS portion of the DBLOGIN command (depending on the database).

owner.table

The owner and name of the checkpoint table. An owner and name are not required
if
they are the same as those specified with the CHECKPOINTTABLE parameter in
the GLOBALS file.

Example

INFO CHECKPOINTTABLE ggs.fin_check
------------------------------------------------------------------
UPGRADE CHECKPOINTTABLE

UPGRADE CHECKPOINTTABLE

Not valid for Replicat for Java, Oracle GoldenGate Applications Adapter, or
Oracle
GoldenGate Big Data.

Use the UPGRADE CHECKPOINTTABLE command to add a supplemental checkpoint table
when
upgrading Oracle GoldenGate from version 11.2.1.0.0 or earlier.

Syntax

UPGRADE CHECKPOINTTABLE [[container. | catalog.]owner.table]

container. | catalog.

The Oracle pluggable database, if applicable. If this option is omitted, the
catalog or
pluggable database defaults to the one that is associated with the SOURCEDB,
USERID, or
USERIDALIAS portion of the DBLOGIN command (depending on the database).

owner.table

The owner and name of the checkpoint table. An owner and name are not required
if
they are the same as those specified with the CHECKPOINTTABLE parameter in the
GLOBALS file.

Example

UPGRADE CHECKPOINTTABLE ggs.fin_check

------------------------------------------------------------------
###################################
#
#ORACLE TRACE TABLE COMMANDS
#
#
###################################
------------------------------------------------------------------
ADD TRACETABLE
```

```
Use ADD TRACETABLE to create a trace table in the Oracle Database. The trace
table
must reside in the schema of the Oracle GoldenGate Extract user, as configured
with
the USERID or USERIDALIAS parameter. The trace table prevents Replicat
transactions
from being extracted again in a bidirectional synchronization configuration.

Use the DBLOGIN command to establish a database connection before using this
command.

The trace table has the following description:

Name          Null?       Type            Description
GROUP_ID      NOT NULL VARCHAR2(8)  The name of the Replicat group or special
run process.
DB_USER               VARCHAR2(30) The user ID of the Replicat group or special
run process.
LAST_UPDATE           DATE            The timestamp of the transaction.

Syntax

ADD TRACETABLE [[container.]owner.table]

container

The pluggable database, if the database is a multitenant container database
(CDB).

owner.table

Optional, use only to specify a trace table with a name that is different from
the
default of GGS_TRACE. The owner must be the same owner that is specified with
the
USERID or USERIDALIAS parameter in the Extract parameter file.
To use the default name, omit this argument. Whenever possible, use the default
table
name. When using a trace table name other than the default of GGS_TRACE, specify
it
with the TRACETABLE parameter in the Extract and Replicat parameter files.
Record the
name, because you will need it for the parameter files and to view statistics or
delete the table.

Examples

Example 1

The following adds a trace table with the default name of GGS_TRACE.

ADD TRACETABLE

Example 2

The following adds a trace table with a user-defined name of ora_trace.

ADD TRACETABLE ora_trace

------------------------------------------------------------------------
```

```
DELETE TRACETABLE

DELETE TRACETABLE

Use DELETE TRACETABLE to delete a trace table. Use the DBLOGIN command to
establish
a database connection before using this command.

Syntax

DELETE TRACETABLE [[container.]owner.table]

container

The pluggable database, if the database is a multitenant container database
(CDB).

owner.table

The owner and name of the trace table to be deleted. An owner and name are not
required if the owner is the same as that specified with the USERID or
USERIDALIAS
parameter and the trace table has the default name of GGS_TRACE.

Example

DELETE TRACETABLE ora_trace
------------------------------------------------------------------
INFO TRACETABLE

INFO TRACETABLE

Use the INFO TRACETABLE command to verify the existence of the specified trace
table in the local instance of the database. If the table exists, Oracle
GoldenGate displays the name and the date and time that it was created;
otherwise Oracle GoldenGate displays a message stating that the table does
not exist. Use the DBLOGIN command to establish a database connection before
using this command.

Syntax

INFO TRACETABLE [[container.]owner.table]

container

The pluggable database, if the database is a multitenant container database
(CDB).

owner.table

The owner and name of the trace table to be verified. An owner and name are
not required if the owner is the same as that specified with the USERID or
USERIDALIAS parameter and the trace table has the default name of GGS_TRACE.

Example

INFO TRACETABLE ora_trace


------------------------------------------------------------------------
####################################
```

```
#
#PMSRVR COMMANDS
#
#
####################################

------------------------------------------------------------------------
INFO PMSRVR

Use the INFO PMSRVR command to determine whether or not the Oracle GoldenGate
Performance Metrics Server is running. This command is an alias for STATUS
PMSRVR.

Syntax:
INFO PMSRVR

------------------------------------------------------------------------
START PMSRVR

Use the START PMSRVR command to start the Oracle GoldenGate Performance Metrics
Server process.

Syntax:
START PMSRVR

When PMSRVR is started for the first time, the sequence of commands must be:

START PMSRVR
START MANAGER
START *
START JAGENT


Thereafter, the sequence of commands is not important because the datastore is
already
created.

------------------------------------------------------------------------
STATUS PMSRVR

Use the STATUS PMSRVR command to determine whether or not the Oracle
GoldenGate Performance Metrics Server is running. This command is an alias for
INFO PMSRVR.
Syntax:
STATUS PMSRVR


------------------------------------------------------------------------
STOP PMSRVR

Use the STOP PMSRVR command to stop the Oracle GoldenGate Performance Metrics
Server process.

Syntax:
STOP PMSRVR [!]

! (Exclamation point character) Bypasses the prompt that confirms the intent
to stop the Performance Metrics Server.

Examples:
```

```
STOP PMSRVR

STOP PMSRVR !


------------------------------------------------------------------------

####################################
#
#JAGENT COMMANDS
#
#
####################################

------------------------------------------------------------------------
INFO JAGENT

Use the INFO JAGENT command to determine whether or not the Oracle GoldenGate
Monitor
JAgent is running. This command is an alias for STATUS JAGENT. For more
information,
see Administering Oracle GoldenGate Monitor.

Syntax

INFO JAGENT

------------------------------------------------------------------------
START JAGENT

START JAGENT

Use the START JAGENT command to start the Oracle GoldenGate Monitor JAgent
process
in a non-clustered environment. In a Windows cluster, start JAgent from the
Cluster
Administrator. For more information, see Administering Oracle GoldenGate
Monitor.

Syntax

START JAGENT

------------------------------------------------------------------------
STATUS JAGENT

STATUS JAGENT

Use the STATUS JAGENT command to determine whether or not the Oracle GoldenGate
Monitor JAgent is running. This command is an alias for INFO JAGENT. For more
information, see Administering Oracle GoldenGate Monitor.

Syntax

STATUS JAGENT

------------------------------------------------------------------------
STOP JAGENT

STOP JAGENT
```

Use the STOP JAGENT command to stop the Oracle GoldenGate Monitor JAgent process in a non-clustered environment. In a Windows cluster, stop JAgent from the Cluster
Administrator. For more information, see Administering Oracle GoldenGate Monitor.

Syntax

```
STOP JAGENT [ ! ]
```

!
(Exclamation point character) Bypasses the prompt that confirms the intent to stop the JAgent.

Examples

Example 1

```
STOP JAGENT
```

Example 2

```
STOP JAGENT !
```

```
------------------------------------------------------------------------
####################################
#
#HEARTBEAT TABLE COMMANDS
#
#
####################################
------------------------------------------------------------------------
```

ADD HEARTBEATTABLE

Use ADD HEARTBEATTABLE to create the objects necessary to use the automatic heartbeat functionality. This command:

* creates a heartbeat seed table, heartbeat table, and heartbeat history table,

* creates the GG_LAG and GG_LAG_HISTORY views,

* creates the GG_UPDATE_HB_TAB and GG_PURGE_HB_TAB procedures that are called by the scheduler jobs,

* creates the scheduler jobs that periodically update the heartbeat and seed table,
and purge the history table,

* populates the seed table.

The default seed, heartbeat, and history table names are GG_HEARTBEAT_SEED, GG_HEARTBEAT,
and GG_HEARTBEAT_HISTORY respectively. The tables, procedures, and scheduler jobs are
created in the GGSCHEMA mentioned in GLOBALS file. The default names can be overridden
by specifying HEARTBEATTABLE hbschemaname.hbtablename in the GLOBALS file. In this
case, the tables, procedures, and jobs are created in the schema, hbschemaname. The

```
seed and history table are created by appending a _SEED and _HISTORY to the
table,
hbtablename.

This command requires a DBLOGIN. On a CDB database, a PDB login is required.

For Oracle, the ADD HEARTBEATTABLE has to be performed in every PDB that you
want
to generate heartbeats for in CDB mode.

For DB2 LUW, you must set the DB2_ATS_ENABLE property with the db2set
DB2_ATS_ENABLE=yes command.

Syntax

ADD HEARTBEATTABLE
 [, FREQUENCY number in seconds]
 [, RETENTION_TIME number in days] |
 [, PURGE_FREQUENCY number in days]
 [, PARTITIONED]
 [, NOADDTRANDATA]
 [, TARGETONLY]

FREQUENCY

Specifies how often the heartbeat seed table and heartbeat table are updated.
For example, how frequently heartbeat records are generated. The default is
60 seconds.

RETENTION_TIME

Specifies when heartbeat entries older than the retention time in the history
table are purged. The default is 30 days.

PURGE_FREQUENCY

Specifies how often the purge scheduler is run to delete table entries that
are older than the retention time from the heartbeat history. The default
is 1 day.

PARTITIONED

Enables partitioning in the heartbeat history table with intervals of 1 day.

NOADDTRANDATA

Valid for Oracle Database only. Does not enable supplemental logging for the
heartbeat
table and the heartbeat seed table. By default, supplemental logging is enabled
for both
tables.

TARGETONLY

Valid for Oracle Database only. Does not enable supplemental logging on both the
heartbeat seed and heartbeat tables. It does not create a scheduler job for
updating
the heartbeat table.

Examples
```

Example 1

The following command creates default heartbeat tables, procedures, and jobs.

```
ADD HEARTBEATTABLE
```

Example 2

The following command creates the heartbeat tables, procedures, and jobs with custom frequency, retention time, and purge frequency.

```
ADD HEARTBEATTABLE, frequency 120, retention_time 10, purge_frequency 2
```

Example 3

The following command creates the heartbeat tables, procedures and jobs with partitioning enabled in the heartbeat history table.

```
ADD HEARTBEATTABLE, partitioned
```

ALTER HEARTBEATTABLE

Use ALTER HEARTBEATTABLE to alter existing seed, heartbeat, and history table options that you set with ADD HEARTBEATTABLE.

This command requires a DBLOGIN. On a CDB database, a PDB login is required.

Syntax

```
ALTER HEARTBEATTABLE
 [FREQUENCY number in seconds]
 [RETENTION_TIME number in days] |
 [PURGE_FREQUENCY number in days]
 [TARGETONLY | NOTARGETONLY]
```

FREQUENCY

Alter frequency to zero (0) is equivalent to pausing the heartbeat. Heartbeat records can be resumed by altering frequency to a value greater than 0.

RETENTION_TIME

Changes the heartbeat retention time specified, in days.

PURGE_FREQUENCY

Changes the repeat interval, in days, of the purge heartbeat table.

TARGETONLY | NOTARGETONLY

Valid for Oracle Database only. TARGETONLY modifies the existing heartbeat seed and
heartbeat tables by disabling supplemental logging on both tables. It
drops the existing scheduler job for updating the heartbeat table.

Valid for Oracle Database only. NOTARGETONLY modifies existing heartbeat seed and heartbeat tables by enabling supplemental logging on both tables. It
creates a new scheduler job for updating the heartbeat table.

```
Examples

ALTER HEARTBEATTABLE FREQUENCY 60

ALTER HEARTBEATTABLE RETENTION_TIME 30

ALTER HEARTBEATTABLE PURGE_FREQUENCY 1

------------------------------------------------------------------
DELETE HEARTBEATTABLE

Use DELETE HEARTBEATTABLE to delete tables, procedures, schedulers, and views.
This command requires a DBLOGIN. On a CDB database, a PDB login is required.

Syntax

DELETE HEARTBEATTABLE group_name

group_name

The name of the process to be cleaned.


------------------------------------------------------------------
DELETE HEARTBEATENTRY

Use DELETE HEARTBEATENTRY to delete the records in the heartbeat table with
the specified process name either in the incoming or outgoing path columns.
This command required a DBLOGIN. On a CDB database, a PDB login is required.

Syntax

DELETE HEARTBEATENTRY group_name

group_name

The name of the process to be cleaned.

------------------------------------------------------------------
INFO HEARTBEATTABLE

Use INFO HEARTBEATTABLE to display information about the heartbeat tables
and options configured in the database.

This command requires a DBLOGIN. On a CDB database, a PDB login is required.

Syntax

INFO HEARTBEATTABLE

------------------------------------------------------------------
ADD PROCEDURETRANDATA

Use ADD PROCEDURETRANDATA to add supplemental logging for Procedural Replication.

This command requires a DBLOGIN. On a CDB database, a PDB login is required.

Syntax

ADD PROCEDURETRANDATA
```

```
------------------------------------------------------------------
DELETE PROCEDURETRANDATA

Use DELETE PROCEDURETRANDATA to remove supplemental logging for Procedural
Replication.

This command requires a DBLOGIN. On a CDB database, a PDB login is required.

Syntax

DELETE PROCEDURETRANDATA


------------------------------------------------------------------
INFO PROCEDURETRANDATA

Use INFO PROCEDURETRANDATA to display supplemental logging information about
Procedural Replication.

This command requires a DBLOGIN. On a CDB database, a PDB login is required.

Syntax

INFO PROCEDURETRANDATA


------------------------------------------------------------------
!
Use the ! command to execute a previous GGSCI command without modifications. To
modify a command before executing it again, use the FCcommand (see "FC"). To
display a list of previous commands, use the HISTORY command (see "HISTORY").
The ! command without arguments executes the most recent command. Options enable
you to execute any previous command by specifying its line number or a text
substring. Previous commands can be executed again only if they were issued
during the current session of GGSCI, because command history is not maintained
from session to session.

Syntax
! [n | -n | string]

n

Executes the command from the specified GGSCI line. Each GGSCI command line
is sequenced, beginning with 1 at the start of the session.
-n

Executes the command issued n lines before the current line.
string

Executes the last command that starts with the specified text string.
Example 1-1 Examples
Example 1
! 9

Example 2
! -3

Example 3
! sta
```

```
------------------------------------------------------------------
ALLOWNESTED | NOALLOWNESTED

Use the ALLOWNESTED and NOALLOWNESTED commands to enable or disable the use of
nested OBEY files. A nested OBEY file is one that contains another OBEY file.

When you exit your GGSCI session, the next GGSCI session will revert back to
NOALLOWNESTED.

Syntax

ALLOWNESTED | NOALLOWNESTED

ALLOWNESTED

Enables the use of nested OBEY files. The maximum number of nested levels is 16.

NOALLOWNESTED

This is the default. An attempt to run a nested OBEY file in the default mode of
NOALLOWNESTED will cause an error that is similar to the following:

ERROR: Nested OBEY scripts not allowed. Use ALLOWNESTED to allow nested scripts.

------------------------------------------------------------------

####################################
#
#MISCELLANEOUS COMMANDS
#
#
####################################
------------------------------------------------------------------
!
Executes the previous command without modifications.

Syntax

! [ n ]


------------------------------------------------------------------
CREATE SUBDIRS

Use CREATE SUBDIRS when installing Oracle GoldenGate. This command creates the
default directories within the Oracle GoldenGate home directory. Use CREATE
SUBDIRS
before any other configuration tasks..

Syntax:

CREATE SUBDIRS


------------------------------------------------------------------
DEFAULTJOURNAL

Use the DEFAULTJOURNAL command to set a default journal for multiple tables
or files for the ADD TRANDATA command when used with a DB2 for i database,
instead of having to use the JOURNAL keyword. Issue this command before issuing
ADD TRANDATA. Any ADD TRANDATA command used without a journal assumes the
```

journal from DEFAULTJOURNAL. To remove the use of a default journal, use the CLEAR option. To display the current setting of DEFAULTJOURNAL, you can issue the command without arguments.

Syntax

DEFAULTJOURNAL [library/journal] [CLEAR]

library/journal

The native name of the journal that you want to use as the default journal for ADD TRANDATA.

CLEAR

Stops the use of a default journal for ADD TRANDATA.

FC

Use FC to display edit a previously issued GGSCI command and then execute it again. Previous commands are stored in the memory buffer and can be displayed by issuing the HISTORY command (see “HISTORY”).

Displaying Previous Commands

Issuing FC without arguments displays the most recent command. Options enable you to execute any previous command by specifying its line number or a text substring.

Previous commands can be edited only if they were issued during the current GGSCI session, because history is not maintained from one session to another.

Editing Commands

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 use one of the following arguments. Arguments are
not case-sensitive and can be combined:

Argument	Description
i text	Inserts text. For example: GGSCI (SysA) 24> fc 9 GGSCI (SysA) 24> send mgr GGSCI (SysA) 24.. i childstatus GGSCI (SysA) 24> send mgr childstatus
r text	Replaces text. For example: GGSCI (SysA) 25> fc 9 GGSCI (SysA) 25> info mgr GGSCI (SysA) 25.. rextract extjd GGSCI (SysA) 25> info extract extjd
d	Deletes a character. To delete multiple characters, enter an d for each d one. For example: GGSCI (SysA) 26> fc 10

```
           GGSCI (SysA) 26> info extract extjd, detail
           GGSCI (SysA) 26.. ddddddddd
           GGSCI (SysA) 26> info extract extjd

text       Replaces the displayed command with the text that you enter on a one-
for-one
           basis. For example:
            replacement text

           GGSCI (SysA) 26> fc 10
           GGSCI (SysA) 26> info mgr
           GGSCI (SysA) 26.. extract extjd
           GGSCI (SysA) 26> info extract extjd

To execute the command, press Enter twice, once to exit the editor and once to
issue
the command. To cancel an edit, type a forward slash (/) twice.

Syntax

FC [n | -n | string]

n

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

-n

Displays the command that was issued n lines before the current line.

string

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

Examples

Example 1

FC 9

Example 2

FC -3

Example 3

FC sta


------------------------------------------------------------------------
HELP

Use HELP to obtain information about an Oracle GoldenGate command. Without
additional options, HELP returns a list of commands. The command option
restricts the output to the specified command.

Syntax

HELP [command]
```

```
command

The command for which you want help.

Example

HELP add replicat


------------------------------------------------------------------------
HISTORY

Use HISTORY to view a list of the most recently issued GGSCI commands since
the startup of the GGSCI session. You can use the ! command or the FC command
to re-execute a command in the list.

Syntax

HISTORY [n]

n

Returns a specific number of recent commands, where n 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 extjd
6: info extract extjd
7: history


------------------------------------------------------------------------
INFO ALL

Use INFO ALL to display the status and lag (where relevant) for all Manager,
Extract, and Replicat processes on a system. When Oracle Grid Infrastructure
Agents (XAG) Clusterware components are in use, the relevant information is
also displayed. The basic command, without options, displays only online
(continuous) processes. To display tasks, use either INFO ALL TASKS or INFO
ALL ALLPROCESSES.

The Status and Lag at Chkpt(checkpoint) fields display the same process status
and lag as the INFO EXTRACT and INFO REPLICAT commands.

If Replicat is in coordinated mode, INFO ALL shows only the coordinator thread.
To view information about individual threads, use INFO REPLICAT.

Example 1-2 Sample INFO ALL Output
Program     Status     Group     Lag at Chkpt     Time Since Chkpt
MANAGER     RUNNING
EXTRACT     ABENDED EXTCUST 00:00:00     96:56:14
EXTRACT     STOPPED INITDL
EXTRACT     STOPPED INITDBL
```

```
Syntax

INFO ALL [TASKS | ALLPROCESSES]

TASKS

Displays information only for tasks.

ALLPROCESSES

Displays information for online processes and tasks.

Examples

Example 1

INFO ALL TASKS

Example 2

INFO ALL ALLPROCESSES

------------------------------------------------------------------------
INFO MARKER

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

Markers can only be added on a NonStop system, using Oracle GoldenGate for
NonStop for HP NonStop software.

The following is an example of the output.

Processed             Added               Diff     Prog    Group    Node
2017-02-16:14:41:15   2017-02-16:14:41:08 00:00:07 Extract PQACMD   \QAMD
                      GROUPCMD REPLICAT RQACMD CLOSEFILES
2017-02-16:14:41:13   2017-02-16:14:41:08 00:00:05 Extract PQACMD   \QAMD
                      TACLCMD REPLICAT RQACMD FUP PURGEDATA $QA16.QAETAR
Where:

* Processed is the local time that a program processed the marker.

* Added is the local time at which the marker was inserted into the NonStop
audit trails or log trails.

* Diff is the time difference between the Processed and Added values. Diff
  can serve as an indicator of the lag between the user application and
  Extract and Replicat activities.

* Prog shows which process processed the marker, such as GGSCI, Logger,
  Extract or Replicat.

* Group shows the Extract or Replicat group or Logger process that processed
  the marker. N/A is displayed if GGSCI processed the marker.

* Node shows the node where the marker was inserted into the audit trails.

* There might be an additional column if user-defined text was included in
  the ADD MARKER statement.
```

Syntax

INFO MARKER [COUNT number]
COUNT number

Restricts the list to a specified number of the most recent markers.

OBEY

Use OBEY to process a file that contains a list of Oracle GoldenGate commands. OBEY is useful for executing commands that are frequently used in sequence.

You can call one OBEY file from another one. This is called a nested OBEY file. You can nest up to 16 OBEY files. To use nested OBEY files, you must enable the functionality by first issuing the ALLOWNESTED command. See “ALLOWNESTED | NOALLOWNESTED”.

Syntax

OBEY file_name

file_name

The relative or fully qualified path name of the file that contains the list of commands.

Examples

Example 1

OBEY ./mycommands.txt

The preceding command executes a file that looks similar to the following example:

```
add extract fin, tranlog, begin now
add exttrail dirdat/aa, extract fin
add extract hr, tranlog, begin now
add exttrail dirdat/bb, extract hr
start extract *
info extract *, detail
```

Example 2

The following example illustrates a nested OBEY file. Assume an OBEY file named addcmds.txt. Inside this file, there is another OBEY command that calls the OBEY file named startcmds.txt, which executes another set of commands.

```
OBEY ./addcmds.txt

(This OBEY statement executes the following:)

add extract fin, tranlog, begin now add exttrail ggs/dirdat/aa, extract fin add
extract hr, tranlog, begin now add exttrail ggs/dirdat/bb, extract hr add
replicat
fin2, exttrail ggs/dirdat/aa, begin now add replicat hr2, exttrail ggs/
dirdat/bb,
begin now obey ./startcmds.txt

(The nested startcmds.txt file executes the following:)
```

```
start extract *
info extract *, detail
start replicat *
info replicat *, detail


------------------------------------------------------------------
SHELL

Use SHELL to execute shell commands from within the GGSCI interface.

Syntax

SHELL command

command

The system command to execute.

Examples

Example 1

SHELL dir dirprm/*

Example 2

SHELL rm ./dat*

------------------------------------------------------------------
SHOW

Use SHOW to display the Oracle GoldenGate environment.

Syntax

SHOW

Example

The following is sample SHOW output. Additional entries may be displayed,
depending on the database type.

Parameter settings:
SET DEBUG OFF
Current directory: C:\GG_81
Using subdirectories for all process files
Editor: notepad
Reports (.rpt)             C:\GG_81\dirrpt
Parameters (.prm)         C:\GG_81\dirprm
Replicat Checkpoints (.cpr)     C:\GG_81\dirchk
Extract Checkpoints (.cpe)     C:\GG_81\dirchk
Process Status (.pcs)         C:\GG_81\dirpcs
SQL Scripts (.sql)         C:\GG_81\dirsql
Database Definitions (.def)     C:\GG_81\dirdef


------------------------------------------------------------------
VERSIONS

Use VERSIONS to display operating system and database version information. For
```

ODBC connections, the driver version is also displayed. To include database information in the output, issue a DBLOGIN command before issuing VERSIONS to establish a database connection.

Syntax

```
VERSIONS
```

VIEW GGSEVT

Use VIEW GGSEVT to view the Oracle GoldenGate error log (ggserr.log file). This file contains information about Oracle GoldenGate events, such as process startup, shutdown, and exception conditions. This information is recorded in the system error log, too, but viewing the Oracle GoldenGate error log sometimes is more convenient and may retain events further back in time.

The display can be lengthy. To exit the display before reaching the end, use the operating system's standard methods for terminating screen output.

Syntax

```
VIEW GGSEVT
```

Example

The following is sample VIEW GGSEVT output:

```
2017-01-08 11:20:56 GGS INFO 301 GoldenGate Manager for Oracle,
mgr.prm: Command received from GUI (START GGSCI ).
2017-01-08 11:20:56 GGS INFO 302 GoldenGate Manager for Oracle,
mgr.prm: Manager started GGSCI process on port 7840.
2017-01-08 11:21:31 GGS INFO 301 GoldenGate Manager for Oracle,
mgr.prm: Command received from GUI (START GGSCI ).
```

VIEW REPORT

Use VIEW REPORT to view the process report or the discard filet that is generated by Extract or Replicat. Each process generates a new report and discard file upon startup.

Reports and discard files are aged whenever a process starts. Old files are appended with a sequence number, for example finance0.rpt, finance1.rpt, and so forth, or discard0.dsc, discard1.dsc, and so forth. To view old files, use the [n] option. To view the current report or discard file, use the command without the [n] option.

Syntax

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

group_name

The name of the Extract or Replicat group. The command assumes the report file named group.rpt or the discard file named group.dscin the Oracle GoldenGate dirrpt sub-directory.

n

The number of an old report. Report files are numbered from 0 (the most recent) to 9 (the oldest).

file_name

The relative file name if stored in the default location, or the full path name if not stored in the default location.

Examples

Example 1

The following displays an old report file (number 3) for the orders group.

```
VIEW REPORT orders3
```

Example 2

The following displays a specific discard file identified by its file name. Note that the file name has a non-default file extension.

```
VIEW REPORT dirrpt/orders.rpt
```