

Oracle® Database

Oracle GoldenGate Microservices Documentation



(19c)

G17497-01

January 2025

ORACLE®

Copyright © 2024, 2025, Oracle and/or its affiliates.

Primary Author: Oracle Corporation

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, software documentation, data (as defined in the Federal Acquisition Regulation), 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," "commercial computer software documentation," or "limited rights data" 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®, Java, MySQL, and NetSuite 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	xiv
Documentation Accessibility	xiv
Related Information	xiv
Conventions	xiv

1 Concepts

Oracle GoldenGate	1-1
Why Do You Need Oracle GoldenGate?	1-1
When Do You Use Oracle GoldenGate?	1-2
Topologies for Oracle GoldenGate	1-2
Oracle GoldenGate Product Family	1-3
Oracle GoldenGate Microservices Architecture	1-4
Features of Oracle GoldenGate Microservices Architecture	1-4
Access Points for Oracle GoldenGate Microservices	1-5
Admin Client	1-6
REST API	1-6
Components of Oracle GoldenGate Microservices Architecture	1-6
Directories and Variables in Microservices Architecture	1-6
About Deployments	1-9
What is a Deployment?	1-9
Secure Deployment	1-9
Non-Secure Deployment	1-9
Local and Remote Deployments	1-9
About Service Manager	1-10
About Administration Server	1-10
About Distribution Server	1-11
About Receiver Server	1-12
About Target-Initiated Distribution Path	1-13
About Performance Metrics Server	1-13
Components of Data Replication in Oracle GoldenGate	1-14
Types of Data Replication Configurations	1-14
Oracle GoldenGate Processes	1-14

Extract	1-14
Replicat	1-14
Distribution Paths for Data Transport	1-15
Oracle GoldenGate Objects	1-15
Trail Files	1-15
Parameter Files	1-16
Checkpoint Files	1-17

2 Install and Patch

Download Oracle GoldenGate Software	2-1
Verify Certification and System Requirements	2-1
Operating System Requirements	2-2
Memory Requirements	2-2
Disk Requirements	2-3
Disk Requirements for Oracle GoldenGate Installation Files	2-3
Temporary Disk Requirements	2-3
Other Disk Space Considerations	2-4
Network	2-4
Operating System Privileges	2-5
Security and Other Considerations	2-5
Windows Console Character Sets	2-6
Other Operating System Requirements	2-6
Prerequisites	2-7
Setting TNS_ADMIN	2-7
Specifying Oracle Variables on UNIX and Linux Systems	2-7
Specifying Oracle Variables on Windows Systems	2-8
What are the Key Microservices Architecture Directories and Variables?	2-9
Installing Oracle GoldenGate	2-12
Installing Oracle GoldenGate Microservices Architecture	2-12
Performing an Interactive Installation with OUI for MA	2-13
Performing a Silent Installation with OUI	2-14
Integrating Oracle GoldenGate Microservices Architecture into a Cluster	2-15
Post-installation Tasks	2-15
Software Installation Directories and Programs for Oracle GoldenGate	2-15
Installing Patches for Oracle GoldenGate Microservices Architecture	2-17
Downloading Patches for Oracle GoldenGate	2-17
Patching Oracle GoldenGate Microservices Architecture Using OPatch	2-18
Uninstalling the Patch for Oracle and Non-Oracle Databases Using OPatch	2-21
Uninstalling Oracle GoldenGate Microservices Architecture	2-21
Removing Deployments and Service Manager	2-21

Removing Deployments and Service Manager Using Oracle GoldenGate Configuration Assistant	2-21
Using Oracle GoldenGate Configuration Assistant - Silent	2-22
Files to be Removed Manually	2-22
Uninstalling Microservices Architecture with Oracle Universal Installer	2-23

3 Deploy

Add a Deployment	3-1
Using OGGCA Wizard for Deployment	3-1
Start the OGGCA Wizard	3-1
Select Service Manager Options	3-2
Configuration Options	3-3
Deployment Details	3-3
Select Deployment Directories	3-3
Specify Environment Variables	3-4
Service Manager Administrator Account	3-6
Specify Security Options	3-7
Advanced Security Settings	3-8
Sharding Options	3-9
Port Settings	3-9
Replication Settings	3-9
Summary	3-10
Configure Deployment	3-11
Finish	3-12
Add a Deployment to an Existing Service Manager	3-12
Add a Deployment in Silent Mode using OGGCA	3-13
First Access to the Deployment from the Service Manager	3-13
Add Deployment Users from the Service Manager	3-13
Add Deployment Users from the Administration Server	3-15
Manage Deployments from the Service Manager	3-16
Quick Tour of the Service Manager	3-16
How to Start and Stop the Service Manager	3-17
How to Change Deployment Details and Configuration	3-18
How to Interpret the Log Information	3-18
How to Enable and Use Debug Logging	3-18
How to Start and Stop Service Manager and Deployments	3-19
Using Scripts to Start and Stop a Deployment	3-19
Remove a Deployment	3-20
Before Removing the Deployment	3-20
Start OGGCA to Remove Deployment	3-20
Remove the Service Manager	3-21

Start OGGCA to Remove the Service Manager	3-21
Files to be Removed Manually After Removing Deployment	3-21
View and Edit Services Configuration	3-22

4 Prepare

Prepare Oracle Database	4-1
Prepare Database Users and Privileges for Oracle	4-1
Grant User Privileges for Oracle Database 21c and Lower	4-1
Privileges for Capturing from Oracle Data Vault	4-5
Configuring Connections for Integrated Processes	4-6
Configuring Logging Properties	4-6
Enabling Minimum Database-level Supplemental Logging	4-8
Enabling Schema-level Supplemental Logging	4-9
Enabling Table-level Supplemental Logging	4-10
Enabling Oracle GoldenGate in the Database	4-12
Configuring Oracle GoldenGate in a Multitenant Container Database	4-12
Using the Root Container Extract from PDB	4-12
Applying to Pluggable Databases	4-13
Excluding Objects from the Configuration	4-14
Requirements for Configuring Container Databases for Oracle GoldenGate	4-14
Setting Flashback Query	4-15
Managing Server Resources	4-17
Ensuring Row Uniqueness in Source and Target Tables	4-17
Oracle: Supported Data Types, Objects, and Operations for DDL and DML	4-18
Details of Support for Oracle Database Editions	4-18
Details of Support for Oracle Data Types and Objects	4-19
Details of Support for Objects and Operations in Oracle DML	4-23
Details of Support for Objects and Operations in Oracle DDL	4-27
Prepare Oracle GoldenGate	4-31
Oracle GoldenGate Users	4-31
Configure Secure Database Connections from Oracle GoldenGate	4-31
Assigning Credentials to Oracle GoldenGate	4-33
Securing the Oracle GoldenGate Credentials	4-33
Add and Alter Database Credentials	4-34
Before Adding Extract and Replicat Processes	4-34
Add TRANDATA	4-34
Add a Checkpoint Table	4-35
Add Heartbeat Table	4-35

5

Extract

Quick Tour of the Administration Service Overview Page	5-1
About Extract	5-1
Add an Extract	5-2
Using Extract Actions	5-6
Downstream Extract for Downstream Database Mining	5-7
Configure Extract for a Downstream Deployment	5-8
Evaluate Extract Options for a Downstream Deployment	5-8
Prepare the Source Database for the Downstream Deployment	5-8
Prepare the Downstream Mining Database to Receive Online Redo Logs	5-12
Enable Downstream Extract to Work with ADG	5-15
Use Cases for Downstream Mining Configuration	5-17
Case 1: Capture from One Source Database in Real-time Mode	5-17
Case 2: Capture from Multiple Sources in Archive-log-only Mode	5-20
Case 3: Capturing from Multiple Sources with Mixed Real-time and Archive-log-only Mode	5-22
Positioning Extract to a Specific Start Point	5-27

6

Distribute

About Distribution Service and Distribution Path	6-1
Distribution Path Streaming Protocols	6-2
Add a Distribution Path	6-2
Add a Target-Initiated Distribution Path	6-7
Manage Distribution Paths	6-13
Manage Distribution Paths	6-13
Reposition a Path	6-13
Change the Path Filtering	6-14
Review the Distribution Path Information	6-17

7

Replicat

Quick Tour of the Administration Service Overview Page	7-1
About Replicat	7-1
Types of Replicat	7-2
About Classic or Non-Integrated Replicat	7-2
About Coordinated Replicat	7-3
About Barrier Transactions	7-4
How Barrier Transactions are Processed	7-4
About Integrated Replicat	7-5
Benefits of Integrated Replicat	7-7
Integrated Replicat Requirements	7-7

About Parallel Replicat	7-7
Benefits of Parallel Replicat	7-8
Parallel Replication Architecture	7-9
Basic Parameters for Parallel Replicat	7-9
Select a Replicat Type for the Deployment	7-10
Add a Replicat	7-15
Basic Parameters for Parallel Replicat	7-17
Additional Parameters for Integrated Replicat	7-18
Example: Add a Nonintegrated Parallel Replicat Using Admin Client	7-20
Using Replicat Actions	7-20
Review Critical Events	7-21

8 Instantiate

About Instantiating with Initial Load Extract	8-1
Add Initial Load Extract Using the Admin Client	8-2
Step 1: Create a Primary Extract	8-2
Step 2: Determine the Instantiation SCN	8-4
Step 3: Create and Start the Initial Load Replicat	8-5
Step 4: Create and start the Initial Load Extract	8-6
Step 5: Create the Distribution Paths	8-7
Step 6: Create the Primary Replicat	8-8

9 Administer

Data Management	9-1
Oracle: DDL Replication	9-1
Prerequisites for Configuring DDL	9-1
Overview of DDL Synchronization	9-1
Limitations of Oracle GoldenGate DDL Support	9-2
Guidelines for Configuring DDL Replication for Oracle	9-4
Understanding DDL Scopes	9-6
Correctly Identifying Unqualified Object Names in DDL	9-8
Enabling DDL Support	9-9
Filtering DDL Replication	9-9
Special Filter Cases	9-11
How Oracle GoldenGate Handles Derived Object Names	9-11
Using DDL String Substitution	9-14
Controlling the Propagation of DDL to Support Different Topologies	9-15
Add Supplemental Log Groups Automatically	9-17
Removing Comments from Replicated DDL	9-17
Replicating an IDENTIFIED BY Password	9-17

How DDL is Evaluated for Processing	9-18
Viewing DDL Report Information	9-19
Tracing DDL Processing	9-21
Procedural Replication	9-21
About Procedural Replication	9-21
Procedural Replication Process Overview	9-22
Determining Whether Procedural Replication Is On	9-22
Enabling and Disabling Supplemental Logging	9-23
Filtering Features for Procedural Replication	9-24
Handling Procedural Replication Errors	9-25
Listing the Procedures Supported for Oracle GoldenGate Procedural Replication	9-25
Monitoring Oracle GoldenGate Procedural Replication	9-26
Execute Commands, Stored Procedures, and Queries with SQLEXEC	9-27
Performing Processing with SQLEXEC	9-27
Using SQLEXEC	9-27
Apply SQLEXEC as a Standalone Statement	9-27
Apply SQLEXEC within a TABLE or MAP Statement	9-28
Using Input and Output Parameters	9-30
Handling SQLEXEC Errors	9-32
Additional SQLEXEC Guidelines	9-33
Set up and Use the Master Keys and Encryption Keys	9-34
Access the Parameter Files	9-34
Configure an Encryption Profile	9-35
Access Extract and Replicat Log Information	9-36
Mapping and Manipulating Data	9-36
Guidelines for Using Self-describing Trails	9-37
Parameters that Control Mapping and Data Integration	9-37
Mapping between Dissimilar Databases	9-37
Globalization Considerations when Mapping Data	9-38
Mapping Columns Using TABLE and MAP	9-41
Configuring Global Column Mapping with COLMATCH	9-44
Understanding Default Column Mapping	9-46
Data Type Conversions	9-47
Selecting and Filtering Rows	9-48
Retrieving Before and After Values	9-53
Selecting Columns	9-54
Selecting and Converting SQL Operations	9-54
Using Transaction History	9-55
Testing and Transforming Data	9-56
Using Tokens	9-62
Bi-Directional Replication	9-63
Prerequisites for Bidirectional Replication	9-64

MySQL: Bi-Directional Replication	9-69
PostgreSQL: Bi-Directional Replication	9-70
Preparing DBFS for an Active-Active Configuration	9-70
Using Edition-Based Redefinition	9-75
Error Management	9-76
Automatic Conflict Detection and Resolution	9-76
About Automatic Conflict Detection and Resolution	9-77
Configuring Delta Conflict Detection and Resolution	9-86
Managing Automatic Conflict Detection and Resolution	9-89
Monitoring Automatic Conflict Detection and Resolution	9-92
Manual Conflict Detection and Resolution	9-95
Overview of the Oracle GoldenGate CDR Feature	9-96
Configuring the Oracle GoldenGate Parameter Files for Error Handling	9-96
Configuring the Oracle GoldenGate Parameter Files for Conflict Resolution	9-101
Making the Required Column Values Available to Extract	9-101
Viewing CDR Statistics	9-102
CDR Example 1: All Conflict Types with USEMAX, OVERWRITE, DISCARD	9-103
CDR Example 2: UPDATEROWEXISTS with USEDELTA and USEMAX	9-109
CDR Example 3: UPDATEROWEXISTS with USEDELTA, USEMAX, and IGNORE	9-111
Error Handling	9-114
Overview of Oracle GoldenGate Error Handling	9-114
Handling Extract Errors	9-114
Handling Replicat Errors during DML Operations	9-115
Handling Replicat Errors during DDL Operations	9-118
Handling TCP/IP Errors	9-118
Maintaining Updated Error Messages	9-119
Resolving Oracle GoldenGate Errors	9-119
Trail File Management	9-119
Manage Trail Files	9-119
Assign Storage for Oracle GoldenGate Trails	9-119
Estimate Space for the Trails	9-120
Add a Trail	9-120
Automate Maintenance Tasks	9-121
Admin Client Command Line Interface for Oracle GoldenGate Microservices	9-122
About Admin Client	9-122
Using Wildcards in Command Arguments	9-125
Using Command History	9-125
Storing and Calling Frequently Used Command Sequences	9-125
Controlling Extract and Replicat	9-126
Deleting Extract and Replicat	9-127
Specifying Object Names in Oracle GoldenGate Input	9-128
Specifying Filesystem Path Names in Parameter Files on Windows Systems	9-128

Supported Database Object Names	9-128
Specifying Names that Contain Slashes	9-130
Qualifying Database Object Names	9-130
Specifying Case-Sensitive Database Object Names	9-132
Using Wildcards in Database Object Names	9-133
Differentiating Case-Sensitive Column Names from Literals	9-135
Creating a Parameter File Using Admin Client	9-135
Creating a Parameter File with a Text Editor	9-137
Simplifying the Creation of Parameter Files	9-137
Using Wildcards	9-137
Using OBEY	9-137
Using Macros	9-138
Using Parameter Substitution	9-138
Validating a Parameter File	9-138
Simplify and Automate Work with Oracle GoldenGate Macros	9-139
Define a Macro	9-140
Call a Macro	9-141
Call a Macro that Contains Parameters	9-142
Call a Macro without Input Parameters	9-144
Calling Other Macros from a Macro	9-145
Create Macro Libraries	9-146
Tracing Macro Expansion	9-147
Using User Exits to Extend Oracle GoldenGate Capabilities	9-147
When to Implement User Exits	9-148
Making Oracle GoldenGate Record Information Available to the Routine	9-148
Creating User Exits	9-149
Supporting Character-set Conversion in User Exits	9-150
Using Macros to Check Name Metadata	9-151
Describing the Character Format	9-151
Upgrading User Exits	9-153
Viewing Examples of How to Use the User Exit Functions	9-153

10 Performance

Monitor	10-1
Commands Used for Monitoring	10-1
Monitor Processes from the Performance Metrics Service	10-4
Review Messages from Messages Tab	10-4
Review Status Changes	10-5
Purge Datastore	10-5
Protocols for Performance Monitoring for Different Operating Systems	10-6
Monitor an Extract Recovery	10-6

Monitor Lag	10-6
About Lag	10-6
Monitor Lag Using Automatic Heartbeat Tables	10-7
Db2 z/OS: Interpret Statistics for Update Operations	10-17
Monitor Processing Volume	10-17
Use the Error Log	10-18
Use the Process Report	10-18
Scheduling Runtime Statistics in the Process Report	10-19
Viewing Record Counts in the Process Report	10-19
Prevent SQL Errors from Filling the Replicat Report File	10-19
Use the Discard File	10-20
Maintain Discard and Report Files	10-20
Parameters Used to Interpret Synchronization Lag	10-21
Tuning	10-21
Tuning the Performance of Oracle GoldenGate	10-21

11 Autonomous Database

About Capturing and Replicating Data Using Autonomous Databases	11-1
Details of Support When Using Oracle GoldenGate with Autonomous Databases	11-1
Configure Extract to Capture from an Autonomous Database	11-3
Establishing Oracle GoldenGate Credentials	11-3
Prerequisites for Configuring Oracle GoldenGate Extract to Capture from Autonomous Databases	11-3
Configure Extract to Capture from an Autonomous Database	11-4
Configure Replicat to Apply to an Oracle Autonomous Database	11-8
Prerequisites for Configuring Oracle GoldenGate Replicat to an Autonomous Database	11-8
Configure Oracle GoldenGate for an Autonomous Database	11-9
Obtain the Autonomous Database Client Credentials	11-9
Configure Replicat to Apply to an Autonomous Database	11-10

12 Upgrade

Obtaining the Oracle GoldenGate Distribution	12-1
Prerequisites	12-1
Oracle GoldenGate Upgrade Considerations	12-1
Extract Upgrade Considerations	12-2
Replicat Upgrade Considerations	12-2
Upgrading Oracle GoldenGate Microservices – GUI Based	12-2
Upgrading Oracle GoldenGate Microservices Using REST APIs	12-3

13 Appendix

Using the LogDump Utility to Access Trail File Records	13-1
Trail Recovery Mode	13-1
Trail Record Format	13-1
Trail File Header Record	13-2
Tokens Area	13-8
Oracle GoldenGate Operation Types	13-8
Checkpoint Tables Additional Details	13-14
Internal Checkpoint Information	13-16
INFO EXTRACT SHOWCH Command: Checkpoint Information	13-17
INFO REPLICAT, SHOWCH: Checkpoint Information	13-19
Supported Character Sets	13-19
Supported Character Sets - Oracle	13-20
Supported Character Sets - Non-Oracle	13-27
Supported Locales	13-35
Commit Sequence Number (CSN)	13-40
Using the Commit Sequence Number	13-41
Connecting Microservices and Classic Architectures	13-43
Connect Oracle GoldenGate Classic Architecture to Microservices Architecture	13-43
Connect Oracle GoldenGate Microservices Architecture to Classic Architecture	13-45

Preface

The *Oracle GoldenGate Microservices Documentation* contains the Oracle GoldenGate Microservices concepts, tasks, advance tasks, security, and other reference information.

Audience

This guide is intended for system administrators and database users to learn about Oracle GoldenGate. It is assumed that readers are familiar with web technologies and have a general understanding of Windows and UNIX platforms.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <https://www.oracle.com/corporate/accessibility/>.

Access to Oracle Support

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

Related Information

- [Oracle GoldenGate Documentation](#)
- [Oracle GoldenGate Veridata](#)

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.
<i>italic</i> <i>italic</i>	Italic type indicates placeholder variables for which you supply particular values, such as in the parameter statement: <code>TABLE <i>table_name</i></code> . 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.

Convention	Meaning
UPPERCASE	Uppercase in the regular text font indicates the name of a process or utility unless the name is intended to be a specific case. Keywords in upper case (ADD EXTRACT, ADD EXTTRAIL, FORMAT RELEASE).
LOWERCASE	Names of processes to be written in lower case. Examples: ADD EXTRACT exte, ADD EXTRAIL ea.
{ }	Braces within syntax enclose a set of options that are separated by pipe symbols, one of which must be selected, for example: { <i>option1</i> <i>option2</i> <i>option3</i> }.
[]	Brackets within syntax indicate an optional element. For example in this syntax, the SAVE clause is optional: CLEANUP REPLICAT <i>group_name</i> [, SAVE <i>count</i>]. Multiple options within an optional element are separated by a pipe symbol, for example: [<i>option1</i> <i>option2</i>].
Sample Locations	Compass directions such as east, west, north, south to be used for demonstrating Extract and Replicat locations. Datacenters names to use the standard similar to dc1, dc2.
Group names	Prefixes for each process, as follows: <ul style="list-style-type: none"> • Extract: ext. Usage with location: extn, where n indicates 'north' compass direction. • Replicat: rep. Usage with location: repn, where n indicates 'north' compass direction. • Distribution Path: dp. Usage with location: dpn, where n indicates 'north' compass direction. • Checkpoint table: ggs_checkpointtable • Trail file names: e or d depending on whether the trail file is for the Extract or distribution path. Suffix derived in alphabetical order. Usage for an Extract trail file: ea, eb, ec. • Trail file subdirectory: The name will use compass directions to refer to the trail subdirectories. Example for trail subdirectory name would be / east, /west, /north, /south.

1

Concepts

Learn about the concepts of Oracle GoldenGate, its components, and Microservices Architecture.

Oracle GoldenGate

Oracle GoldenGate is an application that provides real-time data integration, data replication, transactional change data capture, data transformations, high availability solutions, and verification between operational and analytical enterprise systems.

With Oracle GoldenGate, you can move committed transactions across multiple systems in your enterprise over a secure or non-secure configuration. It supports a wide range of databases and data sources, providing replication between same types or between different databases.

For example, you could replicate between an Oracle Autonomous Database instance and an Oracle Database instance, or between two Oracle Database instances set up as source and target, or a two-way replication between MySQL database and Oracle Database instances. In addition, you can replicate to Java Messaging Queues, flat files, and to Big Data in combination with Oracle GoldenGate for Big Data.

To learn more about Oracle GoldenGate products, see <https://www.oracle.com/integration/goldengate/>.

Why Do You Need Oracle GoldenGate?

Enterprise data is typically distributed across the enterprise in heterogeneous databases. To get data between different data sources, you can use Oracle GoldenGate to load, distribute, and filter transactions within your enterprise in real-time and enable migrations between different databases in near zero-downtime.

To do this, you need a means to effectively move data from one system to another in real-time and with zero-downtime. Oracle GoldenGate is Oracle's solution to replicate and integrate data.

In a data replication environment, Oracle GoldenGate performs the following functions:

- Data movement in real-time, reducing latency.
- Only committed transactions are moved, to leverage consistency and improved performance.
- REST-based microservices to handle different types of data replication environments.
- High performance with minimal overhead on the underlying databases and infrastructure.
- Integration with a wide range of databases providing complete support for replication across different data types, database objects and other requirements.
- Security configurations at different levels and different topologies for a customized secure configuration.

When Do You Use Oracle GoldenGate?

Oracle GoldenGate meets almost any data movement requirement. Some of the most common use cases include:

Business Continuity and High Availability

Business Continuity is the ability of an enterprise to provide its functions and services without any lapse in its operations. High Availability is the highest possible level of fault tolerance. To achieve business continuity, systems are designed with multiple servers, multiple storage, and multiple data centers to provide high availability that supports business continuity in true sense. To establish and maintain such an environment, data needs to be moved between these multiple servers and data centers, which is easily done using Oracle GoldenGate.

Consider a scenario where you are working in a multinational bank that has its headquarters in London, UK. You work in one of the banks' branches in Bangalore, India. This bank uses a specific account for its financial application that is used globally at all the branches. You have been asked by your manager to daily synchronize the transactions that have happened for this account in the database in the Bangalore branch with the centralized database situated at the UK. The volume of transactions is massive, and even the slightest delay can greatly impact the business. This same process is required at multiple destinations for every database in all the branches of the bank worldwide. This process has to be monitored continuously, preferably through some sort of GUI-based tool for the ease of management. Additionally, the bank has several other, non-critical applications used at all the branches. These applications are based on heterogeneous databases, such as MySQL, but the transactions done over these databases also must be loaded into an Oracle Database located at the headquarters. The replication technology used must support both Oracle and heterogeneous databases so that they can talk to each other. Oracle GoldenGate is an apt solution in such a scenario.

Initial Load and Database Migration

Initial load is a process of extracting data records from a source database and loading those records onto a target database. Initial load is a data migration process that is performed only once. Oracle GoldenGate allows you to perform initial load data migrations without taking your systems offline.

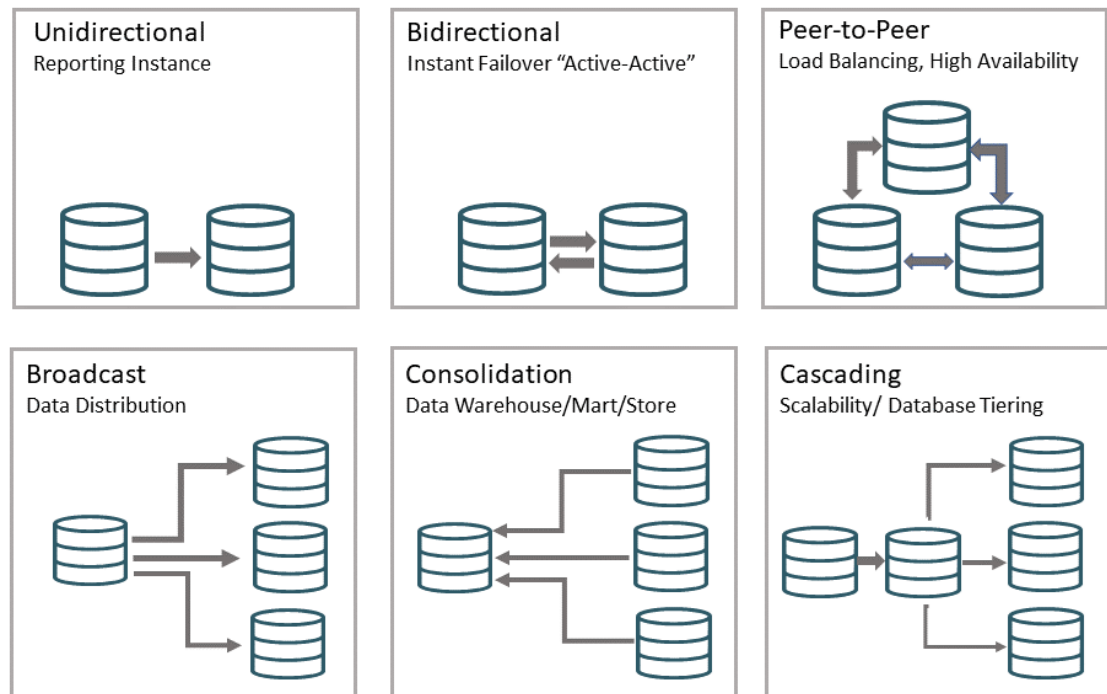
Data Integration

Data integration involves combining data from several disparate sources, which are stored using various technologies, and provide a unified view of the data. Oracle GoldenGate provides real-time data integration.

Topologies for Oracle GoldenGate

After installation, Oracle GoldenGate can be configured to meet your organization's business requirements.

Oracle GoldenGate can be configured in different topologies, ranging from simple unidirectional topology to more complex peer-to-peer. Supported topologies depend on the underlying database requirements and its supported configurations.



Oracle GoldenGate Product Family

There are a wide range of products in the Oracle GoldenGate product family.

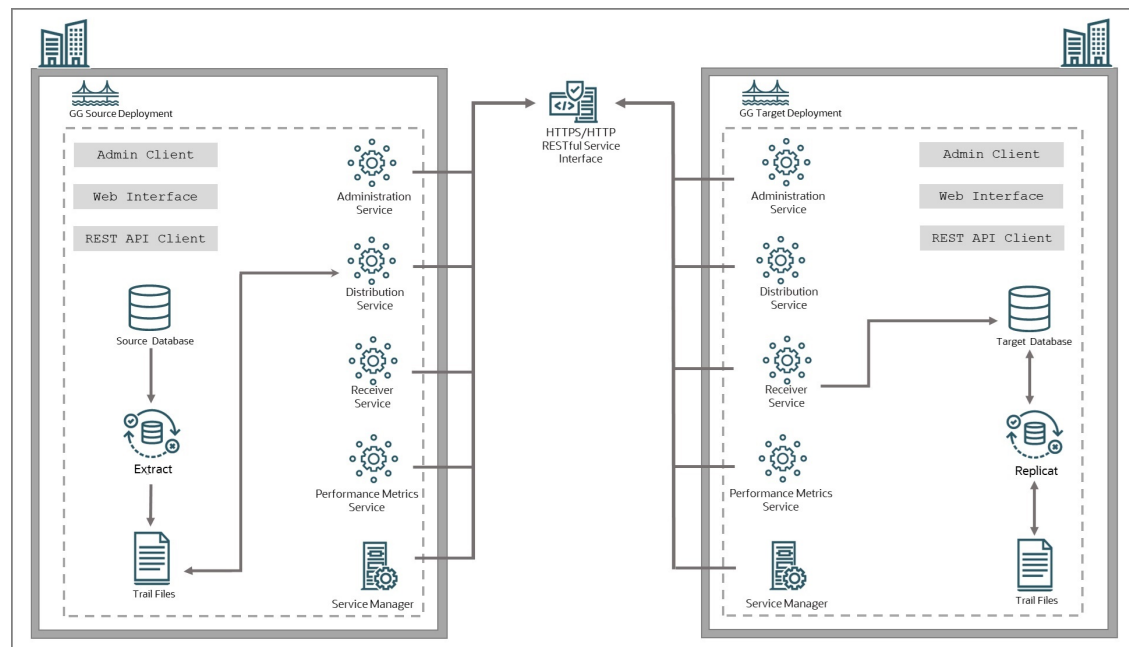
- **Oracle GoldenGate for Oracle Database:** Oracle GoldenGate Microservices for Oracle database provides all the data replication features of Oracle GoldenGate with the features of Oracle Database.
- **Oracle GoldenGate for Non-Oracle Databases:** Oracle GoldenGate Microservices for Oracle database provides all the data replication features of Oracle GoldenGate with all the supported databases including Db2 for i, Db2 z/OS, Db2 LUW, MySQL, PostgreSQL, SQL Server, Sybase, Oracle TimesTen, Teradata.
- **OCI GoldenGate:** Oracle Cloud Infrastructure GoldenGate is a fully managed, native cloud service that moves data in real-time, at scale. OCI GoldenGate processes data as it moves from one or more data management systems to target databases. You can also design, run, orchestrate, and monitor data replication tasks without having to allocate or manage any compute environments.
- **Oracle GoldenGate Free:** Oracle GoldenGate Free provides all the features of the licensed Oracle GoldenGate product, as well as a recipe-driven user interface to easily create and manage replication pipelines. GoldenGate Free deploys from a Docker container onto laptops, on-premises, or any cloud, for free.
- **Oracle GoldenGate Microservices for Marketplace:** Oracle GoldenGate Microservices on Marketplace allows you to deploy Oracle GoldenGate in an off-box architecture, which means you can run and manage your Oracle GoldenGate deployment from a single location.
- **Oracle GoldenGate for HP NonStop (Guardian):** Oracle GoldenGate for HP NonStop enables you to manage business data at a transactional level by extracting and replicating selected data records and transactional changes across a variety of heterogeneous applications and platforms.

- **Oracle GoldenGate Veridata:** Oracle GoldenGate Veridata compares one set of data to another and identifies data that is out-of-sync, and allows you to repair any out-of-sync data.
- **Oracle GoldenGate for Distributed Applications and Analytics:** Oracle GoldenGate for Distributed Applications and Analytics includes Oracle Transaction Manager for Microservices Enterprise Edition, GoldenGate handlers for Big Data, NoSQL, Messaging, Data Warehouse and Data Lakehouse.
- **Oracle GoldenGate Plug-in for EMCC:** The Enterprise Manager Plug-in for Oracle GoldenGate extends the Oracle Enterprise Manager Cloud Control and provides visual support for monitoring and managing Oracle GoldenGate processes.

Oracle GoldenGate Microservices Architecture

Oracle GoldenGate Microservices Architecture (MA) allows you to configure and manage data replication over homogeneous or heterogeneous database environments using RESTful services. These microservices can be accessed using various interfaces including a web interface, command line interface, REST API, or any other service that allows accessing REST-based microservices.

The following diagram illustrates the replication process cycle within a secure (HTTPS) or non-secure (HTTP) environment.



Features of Oracle GoldenGate Microservices Architecture

Oracle GoldenGate Microservices Architecture handles different tasks performed at different stages of the data replication cycle. Some of the product features include the following:

- Oracle GoldenGate Microservices Architecture is bundled with utilities required to configure microservices associated with each deployment. See [Components of Oracle GoldenGate Microservices Architecture](#).

- It is designed with the industry-standard HTTPS communication protocol and the JavaScript Object Notation (JSON) data interchange format.
- The architecture provides options to secure the data replication environment with a variety of security strategies including securing data at rest and in motion, TLS encryption, OAuth 2.0 authentication and authorization, integration with external user authentication services among others.

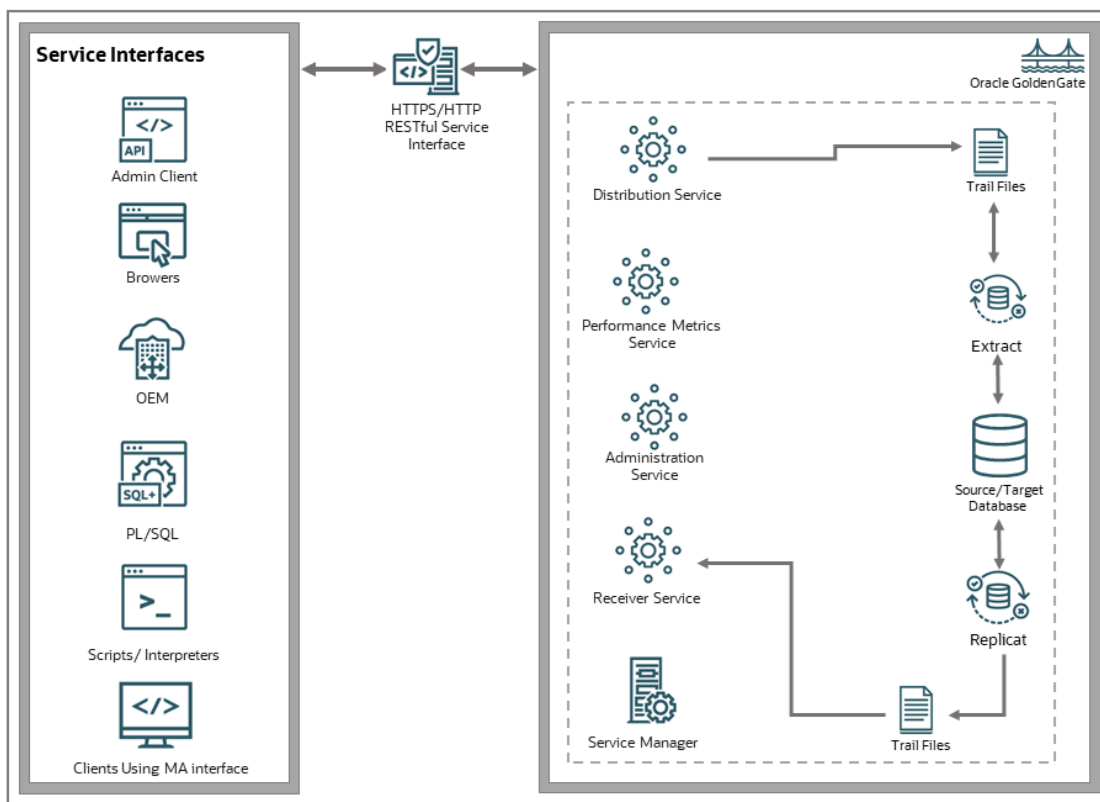
Access Points for Oracle GoldenGate Microservices

Oracle GoldenGate microservices are accessible from a variety of clients or service interfaces. You can use these service interfaces to connect or log in to the microservices and set up data replication tasks, manage and monitor processes using statistical data, tune performance, configure security options, and many other associated tasks.

The Oracle GoldenGate Microservices Architecture is bundled with the following service interfaces:

- Admin Client: Provides access to microservices from the command line.
- Browser-based user interface for a GUI-based experience
- REST API service endpoints

The following diagram shows a variety of clients (Oracle products, command line interface, browsers, and programmatic REST API interfaces) that you can use to access and manage deployments, microservices, and all other Oracle GoldenGate processes.



Microservices Architecture includes an HTML5 based web interface to administer, manage, monitor, and secure deployments. You can access this web interface with the help of URLs specific to each microservice and the Service Manager. The web interface includes the Service

Manager, Administration Service, Distribution Service, Receiver Service, and the Performance Monitoring Service.

You can use these web interface access points to create and run all Extract, Replicat, and Distribution path processes. Along with this, you can set up database credentials, add users that can access the deployment after defining roles for them, and monitor the performance of processes.

See [About Extract](#) and [About Replicat](#).

The REST API provides service endpoints to manage Oracle GoldenGate deployments and replication services. See [REST API Documentation](#).

You can use any of these options to work with your Oracle GoldenGate Microservices Architecture setup.

Admin Client

The Admin Client is a command line utility (similar to the classic GGSCI utility). You can use it to issue the complete range of commands that configure, control, and monitor Oracle GoldenGate. See [About Admin Client](#).

Admin Client is used to create, modify, and remove processes, instead of using the MA web user interface. The Admin Client program is located in the `$OGG_HOME/bin` directory, where `$OGG_HOME` is the Oracle GoldenGate home directory. If you need to automate the Admin Client connection with the deployment, you can use an Oracle Wallet to store the user credentials. The credentials stored must have the following characteristics:

- Single user name (account) and password
- Local to the environment where the Admin Client runs
- Available only to the currently logged user
- Managed by the Admin Client
- Referenced using a credential name
- Available for Oracle GoldenGate deployments and proxy connections.

REST API

The REST API for Oracle GoldenGate provides service endpoints that you use to perform various data replication tasks directly from the REST API interface. This is an alternative to using the web interface or the command line to set up data replication processes and tasks.

See [REST API Documentation](#).

Components of Oracle GoldenGate Microservices Architecture

Directories and Variables in Microservices Architecture

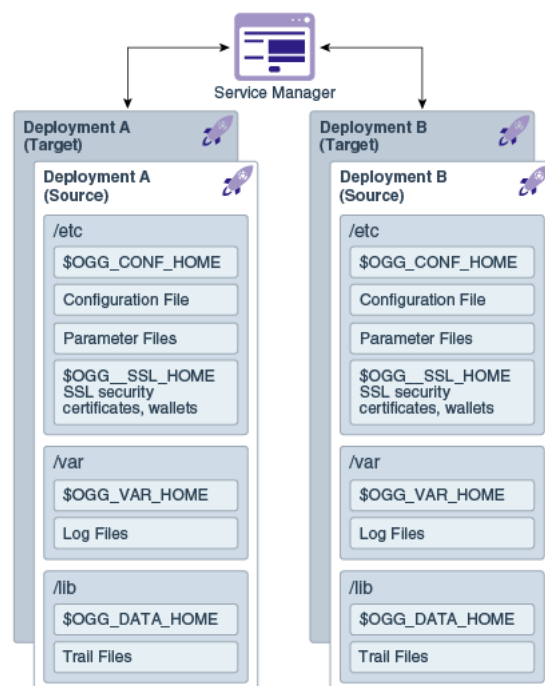
The Microservices Architecture is designed with a simplified installation and deployment directory structure.

This directory structure is based on the Linux Foundation Filesystem Hierarchy Standard. Additional flexibility has been added to allow parts of the deployment subdirectories to be placed at other locations in the file system or on other devices, including shared network devices. The design comprises a read-only Oracle GoldenGate home directory where Oracle

GoldenGate Microservices Architecture is installed and custom deployment specific directories are created as follows:

- bin
- cfgtoollogs
- deinstall
- diagnostics
- include
- install
- inventory
- jdk
- jlib
- lib
 - instantclient
 - sql
 - utl
- OPatch
- oraInst.loc
- oui
- srvm

The following figure shows the files and directories under the Services Manager (srvm) directory:



The following table describes the key MA directories and the variables that are used when referring to those directories during an Oracle GoldenGate installation. When you see these variables in an example or procedure, replace the variable with the full path to the corresponding directory path in your enterprise topology.

Directory Name	Variable	Description	Default Directory Path
Oracle GoldenGate home	OGG_HOME	The Oracle GoldenGate home that is created on a host computer is the directory that you choose to install the product. This read-only directory contains binary, executable, and library files for the product.	/ogg_install_location
Deployment etc home	OGG_ETC_HOME	The location where your deployment configuration files are stored including parameter files.	/ogg_deployment_location/etc
Deployment configuration home	OGG_CONF_HOME	The location where each deployment information and configuration artifacts are stored.	/ogg_deployment_location/etc/conf
Deployment security home	OGG_SSL_HOME	The location where each deployment security artifacts (certificates, wallets) are stored.	/ogg_deployment_location/etc/ssl
Deployment variable home	OGG_VAR_HOME	The location where each deployment logging and reporting processing artifacts are stored.	/ogg_deployment_location/var
Deployment data home	OGG_DATA_HOME	The location where each deployment data artifacts (trail files) are stored.	/ogg_deployment_location/var/lib/data

You can change the default location of all of these to customize where you want to store these files.

In a configuration where the `OGG_VAR_HOME` is a local directory and the `OGG_HOME` is a shared read-only remote directory, many deployments with local `OGG_VAR_HOME` can share one read-only shared `OGG_HOME`.

This directory design facilitates a simple manual upgrade. To upgrade, you stop the services and then set the `OGG_HOME` in the web interface (or via a REST command) and then restart the processes. On restart, Oracle GoldenGate picks up the updated environment variables. You simply switch a deployment to use a new Oracle GoldenGate release by changing the `Oracle GoldenGate home` directory path in the Service Manager to a new Oracle GoldenGate home directory, which completes the upgrade. Restart the microservices, Extract and Replicat processes.

About Deployments

A deployment is a configuration to set up for Oracle GoldenGate Microservices to allow creating users, choose if you want to create a secure SSL environment, define the host and port for various microservices offered with Oracle GoldenGate Microservices Architecture. When you add a deployment for the first time, you can set up a new Service Manager and then add more deployments to the existing Service Manager.

What is a Deployment?

A deployment is a configuration package to set up Oracle GoldenGate Microservices for your choice of database. Deployments can be setup to be secure or non-secure and are added to a Service Manager.

Oracle GoldenGate Configuration Wizard (OGGCA) is a utility that allows you to configure your deployments. See the [Add a Deployment](#) topic to learn more about using OGGCA to configure various options associated with a deployment.

When you start the deployment configuration for the first time on the host server:

- Decide if you need a secure or non-secure deployment. This is because you cannot change from secure to non-secure or non-secure to secure deployments after configuration.
- Configure a new Service Manager on your host server. After the first time configuration, all new deployments should be added to the existing Service Manager available on the host server.



Note:

Oracle recommends that you have a single Service Manager per host server, to avoid redundant upgrade and maintenance tasks with Oracle GoldenGate releases.

Secure Deployment

If you decide to set up a secure deployment, then the deployment configuration provides you options to set up a secure SSL/TLS connection, using server and client certificates.

A secure deployment uses RESTful API calls over an SSL/TLS connection to transmit trail data between the Distribution Service and Receiver Service.

See [Specify Security Options](#) in the [Add a Deployment](#) topic, to learn about configuring security for source and target deployments.

Non-Secure Deployment

For a non-secure deployment, you don't need to apply server and client side certificates for the deployment. RESTful API calls occur over plain-text HTTP over the network.

Local and Remote Deployments

- **Local deployment:** For a local deployment, the source database and Oracle GoldenGate are installed on the same server. No extra consideration is needed for local deployments.

- **Remote deployment:** For a remote deployment, the source database and Oracle GoldenGate are installed on separate servers.

About Service Manager

A Service Manager acts as a watchdog for other services available with Microservices Architecture.

A Service Manager allows you to manage one or multiple Oracle GoldenGate deployments on a local host. A Service Manager has a one to many relationship with the Administration Service. Each Oracle GoldenGate installation has a single Service Manager that is responsible for multiple deployments.

Optionally, Service Manager may run as a system service and maintains inventory and configuration information about your deployments and allows you to maintain multiple local deployments. Using Service Manager, you can start and stop instances, and query deployments and the other services.

See [Manage Deployments from the Service Manager](#).

About Administration Server

The Administration Server supervises, administers, manages, and monitors processes within an Oracle GoldenGate deployment.

The Administration Server operates as the central control entity for managing the replication components in your Oracle GoldenGate deployments. You use it to create and manage your local Extract and Replicat processes without the need to access the server where Oracle GoldenGate is installed. The key feature of the Administration Server is the REST API service Interface that can be accessed from any HTTP or HTTPS client, such as the Microservices Architecture service interfaces or other clients like Perl and Python.

In addition, the Admin Client can be used to make REST API calls to communicate directly with the Administration Server. See [Admin Client](#) for details.

The Administration Server is responsible for coordinating and orchestrating Extracts, Replicats, and paths to support greater automation and operational managements. Its operation and behavior is controlled through published query and service interfaces. These interfaces allow clients to issue commands and control instructions to the Administration Server using REST JSON-RPC invocations that support REST API interfaces.

The Administration Server includes an embedded web application that you can use directly with any web browser and does not require any client software installation.

Use the Administration Server to create and manage:

- Extract and Replicat processes
 - Add, alter, and delete
 - Register and unregister
 - Start and stop
 - Review process information, statistics, reports, and status including LAG and checkpoints
 - Retrieve the report and discard files
- Configuration (parameter) files
- Checkpoint, trace, and heartbeat tables

- Supplemental logging for procedural replication, schema, and tables
- Tasks both custom and standard, such as auto-restart and purge trails
- Credential stores
- Encryption keys (`MASTERKEY`)
- Add users and assign their roles

About Distribution Server

Distribution Server functions as a networked data distribution agent in support of conveying and processing data and commands in a distributed deployment. It is a high performance application that is able to handle multiple commands and data streams from multiple source trail files, concurrently.

Distribution Server replaces the classic multiple source-side data pumps with a single instance service. This service distributes one or more trails to one or more destinations and provides lightweight filtering only (no transformations).

Multiple communication protocols can be used, which provide you the ability to tune network parameters on a per path basis. These protocols include:

- Oracle GoldenGate protocol for communication between the Distribution Server and the Collector in a non services-based (classic) target. It is used for inter-operability.

Note:

TCP encryption does not work in a mixed environment of Classic and Microservices architecture. The Distribution Server in Microservices Architecture cannot be configured to use the TCP encryption to communicate with the Server Collector in Classic Architecture running in a deployment. Also, the Receiver Service in Microservices Architecture cannot accept a connection request from a data pump in Classic Architecture configured with `RMTHOST ... ENCRYPT` parameter running in a deployment.

- WebSockets for HTTPS-based streaming, which relies on SSL security.
- UDP protocols.
- Proxy support for cloud environments:
 - SOCKS5 for any network protocol.
 - HTTP for HTTP-type protocols only, including WebSocket.
- Passive Distribution Server to initiate path creation from a remote site. Paths are source-to-destination replication configurations though are not included in this release.

Note:

Distribution Server cannot filter data in the trail. A distribution path will send all trail data from source to target based on the specified target authentication protocol.

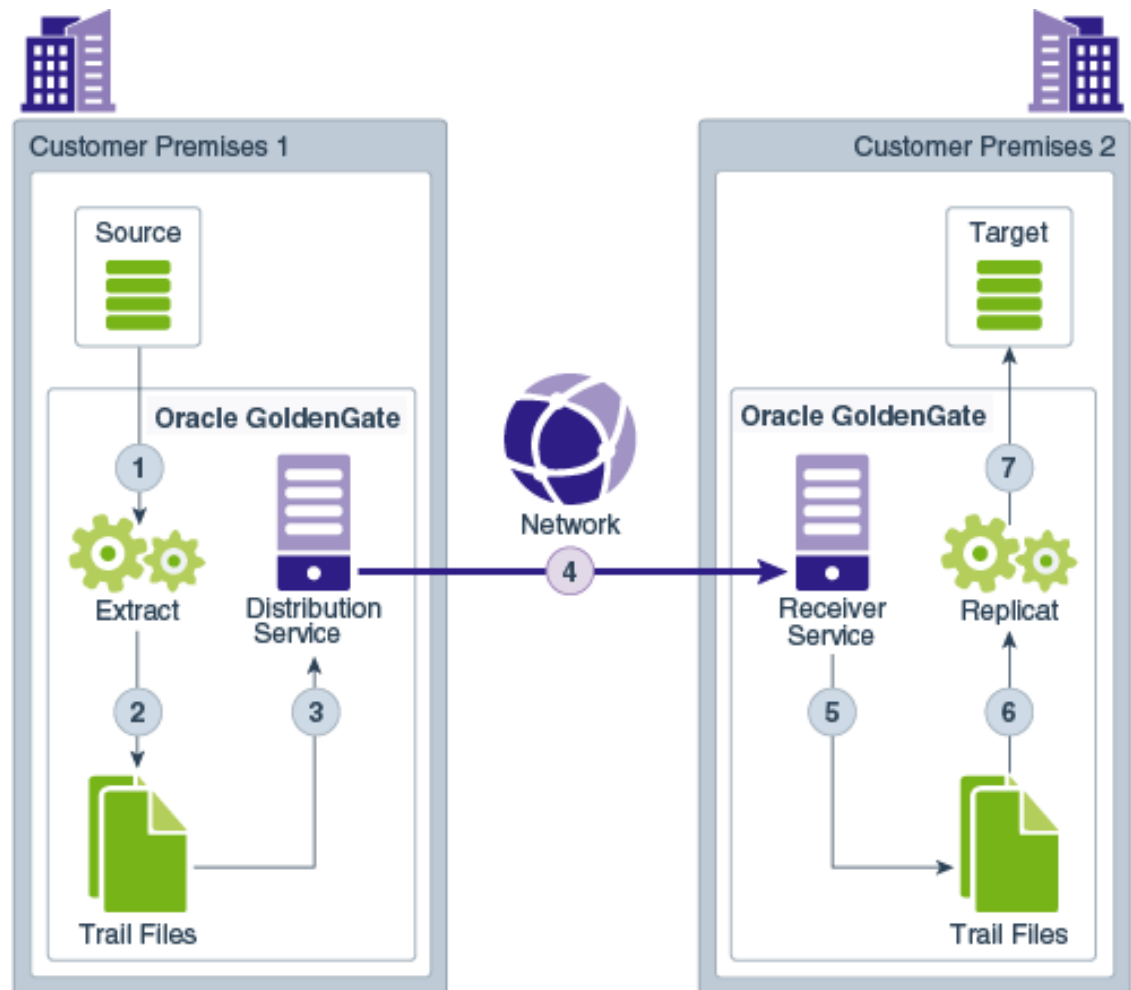
About Receiver Server

A Receiver Server is the central control service that handles all incoming trail files. It interoperates with the Distribution Service and it replaces multiple discrete target-side Collectors with a single instance service.

Use Receiver Server to:

- Monitor path events
- Add target-initiated paths
- Query the status of incoming paths
- View the statistics of incoming paths
- Diagnose path issues

WebSockets (ws) is the default HTTPS initiated full-duplex streaming protocol used by the Receiver Server. It enables you to fully secure your data using SSL security. The Receiver Server seamlessly traverses through HTTP forward and reverse proxy servers.



Additionally, the Receiver Server supports the following protocols:

- UDP-based protocol for wide area networks: For more information, see <http://udt.sourceforge.net/>.
- Classic Oracle GoldenGate protocol for classic deployments so that the Distribution Service communicates with the Collector and the Data Pump communicates with the Receiver Server.

**Note:**

TCP encryption does not work in a mixed environment of Classic and Microservices architecture. The Distribution Service in Microservices Architecture cannot be configured to use the TCP encryption to communicate with the Server Collector in Classic Architecture running in a deployment. Also, the Receiver Server in Microservices Architecture cannot accept a connection request from a data pump in Classic Architecture configured with `RMTHOST ... ENCRYPT` parameter running in a deployment.

:

About Target-Initiated Distribution Path

Target-initiated paths for microservices enable the Receiver Service to initiate a path to the Distribution Service on the target deployment and pull trail files. This feature allows the Receiver Service to create a target initiated path for environments such as Demilitarized Zone Paths (DMZ) or Cloud to on-premise, where the Distribution Service in the source Oracle GoldenGate deployment cannot open network connections in the target environment to the Receiver Service due to network security policies.

If the Distribution Service cannot initiate connections to the Receiver Service, but Receiver Service can initiate a connection to the machine running the Distribution Service, then the Receiver Service establishes a secure or non-secure target initiated path to the Distribution Service through a firewall or Demilitarized (DMZ) zone using Oracle GoldenGate and pull the requested trail files.

The Receiver Service endpoints display that the retrieval of the trail files was initiated by the Receiver Service.

About Performance Metrics Server

All Oracle GoldenGate processes send metrics to the Performance Metrics Server, which enables you to monitor the performance of all processes from a single interface.

The Performance Metrics Server uses metrics to collect and store instance deployment performance results. This metrics collection and repository is separate from the administration layer information collection. You can monitor performance metrics using other embedded web applications and use the data to tune your deployments for maximum performance.

Use the Performance Metrics Server to:

- Query for various metrics and receive responses in the services JSON format or the classic XML format
- Integrate third party metrics tools
- View error logs

- View active process status
- Monitor system resource utilization

Components of Data Replication in Oracle GoldenGate

Types of Data Replication Configurations

Oracle GoldenGate can be configured for the following purposes:

- A static extraction of data records from one database and loading of those records to another database or data source.
- Continuous extraction and replication of transactional Data Manipulation Language (DML) operations and Data Definition Language (DDL) changes (for supported databases) to keep source and target data consistent.
- Data extraction from supported database sources and replication to Big Data and file targets using Oracle GoldenGate Distributed Applications and Analytics.

Oracle GoldenGate Processes

Extract

The Extract process is configured to run on the source endpoint from where the committed database transactions need to be captured. This process is the extraction or the data capture mechanism of Oracle GoldenGate.

You can configure the Extract process to capture data from the following types of data sources:

- **Source tables:** This source type is used for initial loads.
- **Database recovery logs or transaction logs:** While capturing from the logs, the actual method varies depending on the database type. An example of this source type is the Oracle database redo logs.

See [About Extract](#) to learn more.

Replicat

The Replicat process applies the updates from the trail files to the target database. It reads the trail file on the target database, reconstructs the DML or DDL operations, and applies them to the target database.

The Replicat process uses dynamic SQL to compile a SQL statement once and then executes it many times with different bind variables. You can configure the Replicat process so that it waits a specific amount of time before applying the replicated operations to the target database.

For example, a delay may be desirable to prevent the propagation of errant SQL, to control data arrival across different time zones, or to allow time for other planned events to occur.

For the two common uses cases of Oracle GoldenGate, Replicat functions as follows:

- **Initial Loads:** When you set up Oracle GoldenGate for initial loads, the Replicat process applies a static data copy to target objects or routes the data to a high-speed bulk-load utility.

- **Change Synchronization:** When you set up Oracle GoldenGate to keep the target database synchronized with the source database, the Replicat process applies the source operations to the target objects using a native database interface or ODBC, depending on the database type.

You can configure multiple Replicat processes with one or more Extract processes in parallel to increase throughput. To preserve data integrity, each set of process handles a different set of objects. To differentiate among Replicat processes, you can create Replicat groups with a unique group name.

See [About Replicat](#) to learn about different types of Replicats modes.

Distribution Paths for Data Transport

A distribution path or DISTPATH defines the path of trail file between endpoints. The distribution path is configured from the Distribution Service. See [Distribution Service](#) to learn more.

A target-initiated distribution path, which is also called the receiver path or RECVPATH defines the path of the trail, from the Receiver Service to the Distribution Service in environments with secure target endpoints. See [Add a Target-Initiated Distribution Path](#).

Oracle GoldenGate Objects

Trail Files

A trail is a series of files on disk where Oracle GoldenGate stores the captured changes to support the continuous extraction and replication of database changes.

A trail can exist on the source system, an intermediary system, the target system, or any combination of these systems, depending on how you configure Oracle GoldenGate. On the local system, it is known as an Extract trail (or local trail). On a remote system, it is known as a remote trail. By using a trail for storage, Oracle GoldenGate supports data accuracy and fault tolerance. The use of a trail also allows extraction and replication activities to occur independently of each other. With these processes separated, you have more choices for how data is processed and delivered. For example, instead of extracting and replicating changes continuously, you could extract changes continuously and store them in the trail for replication to the target later, whenever the target application needs them.

In addition, trails allow Oracle database to operate in heterogeneous environment. The data is stored in a trail file in a consistent format, so it can be read by the Replicat process for all supported databases.

Processes that Write to the Trail File

Oracle GoldenGate Extract writes to the trail file. All local trails must have different full-path names though you can use the same trail names in different paths.

In Oracle GoldenGate MA, distribution paths and receiver paths are used to distribute remote trails. The Distribution Service and Receive Service are used to configure distribution path and receiver path, respectively. Distribution path transfers the trail over a network, to defined targets. The trail may contain data from multiple Extracts, which transferred to a remote system.

Processes that Read from the Trail File

The Replicat processes, and the Distribution Path read from the trail files. Extract captures DML and DDL operations using a local trail, performs further processing if needed, and transfers the data to a trail that is read by the next Oracle GoldenGate process, which is the Replicat.

In case of distributed deployment, a Distribution Service process will read the remote trail file and send it across the network to a waiting Receiver Service process.

The Replicat process reads the trail and applies the replicated DML and DDL operations to the target database.

Trail File Creation and Maintenance

The trail files are created as needed during processing. You specify a two-character name for the trail when you add it to the Oracle GoldenGate configuration with the `ADD RMTTRAIL` or `ADD EXTTRAIL` command. By default, trails are stored in the `dirdat` sub-directory of the Oracle GoldenGate directory. You can specify a six or nine digit sequence number using the `TRAIL_SEQLEN_9D | TRAIL_SEQLEN_6D GLOBALS` parameter; `TRAIL_SEQLEN_9D` is set by default. It is recommended to use the 9-digit sequence number when possible.

As each new file is created, it inherits the two-character trail name appended with a unique nine digit sequence number from 000000000 through 999999999 (for example `c:\ggs\dirdat\tr000000001`). When the sequence number reaches 999,999,999 or 999,999 (depending on the prior setting) the Extract process will abend.

Trail files can be purged on a routine basis by using the Manager parameter `PURGEOLDEXTRACTS`.

You can create more than one trail to separate the data from different objects or applications. To maximize throughput, and to minimize I/O load on the system, extracted data is sent into and out of a trail in large blocks. The transactional order of the trail file or the trail sequence is preserved.

Parameter Files

Most Oracle GoldenGate functionality is controlled by means of parameters specified in parameter files. A parameter file is a plain text file that is read by an associated Oracle GoldenGate process.

Oracle GoldenGate Microservices Architecture uses the following runtime parameters:

- **Global runtime parameters:** These are different from the `GLOBALS` parameter. They apply to all database objects that are specified in a parameter file. Some global runtime parameters affect process behavior, while others affect such things as memory utilization. `USERIDALIAS` is an example of a global runtime parameter. A global parameter should be listed only once in the file. When listed more than once, only the last instance is active, and all other instances are ignored.
- **Object-specific parameter:** These parameters enable you to apply different processing rules for different sets of database objects. `GETINSERTS` and `IGNOREINSERTS` are examples of object-specific parameters. Each precedes a `MAP` statement that specifies the objects to be affected. Object-specific parameters take effect in the order of their listing in the file.

Runtime parameters allow controlling various aspects of Oracle GoldenGate synchronization, such as:

- Data selection, mapping, transformation, and replication
- DDL and sequence selection, mapping, and replication (where supported)
- Error resolution
- Logging
- Status and error reporting
- System resource usage
- Startup and runtime behavior

Although you can have multiple Extracts and Replicats running in a single deployment, each one can only be associated with a single parameter file. Extracts and Replicats are identified by their case-insensitive name. For example, an Extract called **exte**, would have 1 associated parameter file called **exte.prm**.

See [Working with Parameter Files](#) to learn more.

Checkpoint Files

When database checkpoints are used, Oracle GoldenGate creates a checkpoint table with a user-defined name in the database, using Oracle GoldenGate commands. These checkpoint tables are created for Extract and Replicat processes. For Extract, there are read and write checkpoints set up at data source. For Replicat, the checkpoint is set up in the trail file.

See [Checkpoint Tables Additional Details](#) .

2

Install and Patch

Learn about installation prerequisites for Oracle GoldenGate, steps to install Oracle GoldenGate for different databases, post-installation tasks, installing patches, and uninstalling Oracle GoldenGate.

Download Oracle GoldenGate Software

You can download Oracle GoldenGate from the Oracle GoldenGate Downloads page at <https://www.oracle.com/middleware/technologies/goldengate-downloads.html> and from the Oracle Software Delivery Cloud site, at <https://edelivery.oracle.com/osdc/faces/SoftwareDelivery>.

Verify Certification and System Requirements

Ensure that Oracle GoldenGate is installed on supported hardware and operating systems. For more information, see the [Certification Matrix](#) for the release.

Oracle tests and verifies the performance of your product on all certified systems and environments. As new certifications occur, they are added to the proper certification document. New certifications can occur at any time, and for this reason the certification documents are kept outside of the documentation libraries and are available on Oracle Technology Network.

Here are some additional details about the supported platforms:

- **Cross Endian Support:** Most Oracle GoldenGate products support cross endian replication, which means that the source and target database can be a different platform (or even endian) than the actual server where Oracle GoldenGate is installed.
- **Fully Certified Criteria:** Oracle GoldenGate certifications are often phased in, for a particular new release of the product. Oracle typically supports Oracle databases first and then the various non-Oracle and Big Data technologies. In some cases, Oracle GoldenGate may support the data store you are looking for, but you may need to check the certification matrix for a previous release. Platforms that are in the certification matrix are platforms where either full regression testing is done or where basic validation is performed for continuity purposes.
- **Fully Supported by Inference:** There are other technologies that are supported for Oracle GoldenGate that may not be explicitly listed in the certification matrix. For example, Oracle certify its technologies based on a combination of Chipset, Operating System, Data Store Type, and Data Store Version. As long as these four criteria are met, support is available.
- **Fully Supported through Open Source Compatibility:** There are a number of Open Source technologies that Oracle GoldenGate is certified with such as Big Data and non-Oracle databases. Sometimes, users may have open source environments and need Oracle GoldenGate to provide support with such unique infrastructures, such as Apache HBase on Azure Data Lake. In such cases, Oracle GoldenGate does support any unique open source environment if the Chipset, Operating System, Open Source Framework and Framework Version are certified by Oracle GoldenGate. For example, in case of Apache HBase, Oracle GoldenGate support needs to check the version of Apache HBase, for which Oracle GoldenGate is certified, and if that version happens to be running on some Cloud, then Oracle GoldenGate will be supported. In each of these Open Source examples

(that are not explicitly certified), Oracle GoldenGate support is available using the base open source configurations, such as Apache on certified hardware. However, Oracle may not be obligated to support each possible infrastructure combination that users may select.

- **Java JDBC Support:** Many SQL, NoSQL and Big Data technologies support Java JDBC capabilities. Oracle GoldenGate for Distributed Applications and Analytics enables replication of transactions into any JDBC compliant drivers. Individual drivers may vary in terms of performance and metadata coverage, so there is no specific guarantee that Oracle GoldenGate JDBC support will work with every JDBC driver, but most common JDBC drivers and commercial implementations usually work with Oracle GoldenGate JDBC and these are supported. If you don't find your technology in the certification matrix, but you know that there is a JDBC drive available, then it could be that you may have both technical compatibility and a supported configuration.
- **Managed and Unmanaged Data Stores:** With the advent of managed Cloud services such as native cloud services, many data stores are now available with automated lifecycle, patching, and other conveniences. In many cases, managed data stores are fully compatible and consistent with Oracle GoldenGate certifications and support. However, in some cases, a cloud vendor may turn-off or restrict access to features that Oracle GoldenGate requires for full features compatibility, particularly with Oracle GoldenGate Extract capabilities. If you have a question about a third party cloud managed service for a data store that Oracle GoldenGate may usually support, but you do not see that managed service listed in the Oracle GoldenGate certification matrix, directly contact Oracle GoldenGate product management.

Operating System Requirements

This section outlines the operating system resources that are necessary to support Oracle GoldenGate.

Topics:

Memory Requirements

All Platforms

The amount of memory that is required for Oracle GoldenGate depends on the amount of data being processed, the number of Oracle GoldenGate processes running, the amount of RAM available to Oracle GoldenGate, and the amount of disk space that is available to Oracle GoldenGate for storing pages of RAM temporarily on disk when the operating system needs to free up RAM (typically when a low watermark is reached). This temporary storage of RAM to disk is commonly known as **swapping** or **paging** (herein referred to as swapping). Depending on the platform, the term *swap space* can be a swap partition, a swap file, a page file (Windows) or a shared memory segment (IBM for i).

Modern servers have sufficient RAM combined with sufficient swap space and memory management systems to run Oracle GoldenGate. However, increasing the amount of RAM available to Oracle GoldenGate may significantly improve its performance, as well as that of the system in general.

Typical Oracle GoldenGate installations provide RAM in multiples of gigabytes to prevent excessive swapping of RAM pages to disk. The more contention there is for RAM the more swap space that is used.

Excessive swapping to disk causes performance issues for the Extract process in particular, because it must store data from each open transaction until a commit record is received. If

Oracle GoldenGate runs on the same system as the database, then the amount of RAM that is available becomes critical to the performance of both.

RAM and swap usage are controlled by the operating system, not the Oracle GoldenGate processes. The Oracle GoldenGate cache manager takes advantage of the memory management functions of the operating system to ensure that the Oracle GoldenGate processes work in a sustained and efficient manner. In most cases, users need not change the default Oracle GoldenGate memory management configuration.

For more information about evaluating Oracle GoldenGate memory requirements, see the `CACHEMGR` parameter in the *Parameters and Functions Reference for Oracle GoldenGate*. Also, see *Tuning the Performance of Oracle GoldenGate* in *Administering Oracle GoldenGate*.

Windows Platforms

For Windows Server environments, the number of process groups that can be run are tightly coupled to the *non-interactive* Windows desktop heap memory settings. The default settings for Windows desktop heap may be enough to run very small numbers of process groups. As you approach larger amounts of process groups, more than 60 or so, you have two choices:

- Adjust the non-interactive value of the SharedSection field in the registry based on information from Microsoft (Windows desktop heap memory).
- Increase the number of Oracle GoldenGate homes and spread the total number of desired process groups across these homes.

For more information on modifying the Windows Desktop Heap memory, review the following Oracle Knowledge Base document (Doc ID 2056225.1).

Disk Requirements

Disk space requirements vary based on the platform, database, and Oracle GoldenGate architecture to be installed.

Disk Requirements for Oracle GoldenGate Installation Files

The disk space requirements for a Oracle GoldenGate installation vary based on your operating system and database. Ensure that you have adequate disk space for the downloaded file, expanded files, and installed files, which can be up to 2GB.

Temporary Disk Requirements

When total cached transaction data exceeds the `CACHESIZE` setting of the `CACHEMGR` parameter, Extract begins writing cache data to temporary files located in the Oracle GoldenGate installation directory. For Classic Architecture, this is in the installation's `dirtmp` folder, and for Microservices Architecture, it is the `/var/temp` folder for that deployment.

The cache manager assumes that all of the free space on the file system is available. These directories can fill up quickly if there are many transactions with large transaction sizes. To prevent I/O contention and possible disk-related Extract failures, dedicate a disk to this directory. You can assign a name to this directory with the `CACHEDIRECTORY` option of the `CACHEMGR` parameter.

**Note:**

CACHEMGR is an internally self-configuring and self-adjusting parameter. 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.

It is typically more efficient for the operating system to swap to disk than it is for Extract to write temporary files. The default `CACHESIZE` setting assumes this. Thus, there should be sufficient disk space to account for this, because only after the value for `CACHESIZE` is exceeded will Extract write transaction cached data to temporary files in the file system name space. If multiple Extract processes are running on a system, the disk requirements can multiply. Oracle GoldenGate writes to disk when there is not enough memory to store an open transaction. Once the transaction has been committed or rolled back, committed data is written to trail files and the data are released from memory and Oracle GoldenGate no longer keeps track of that transaction. There are no minimum disk requirements because when transactions are committed after every single operation these transactions are never written to disk.

**Note:**

Oracle recommends that you do not change the `CACHESIZE` because performance can be adversely effected depending on your environment.

Other Disk Space Considerations

In addition to the disk space required for the files and binaries that are installed by Oracle GoldenGate, allow additional disk space to hold the Oracle GoldenGate trails. Trails can be created up to 2GB in size, with a default of 500MB. The space required depends upon the selected size of the trails, the amount of data being captured for replication, and how long the consumed trails are kept on the disk. The recommended minimum disk allocated for Trails may be computed as:

$((\text{transaction log size} * 0.33) * \text{number of log switches per day}) * \text{number of days to retain trails}$

Based on this equation, if the transaction logs are 1GB in size and there is an average of 10 log switches per day, it means that Oracle GoldenGate will capture 3.3GB data per day. To be able to retain trails for 7 days, the minimum amount of disk space needed to hold the trails is 23GB.

A trail is a set of self-aging files that contain the working data at rest and during processing. You may need more or less than this amount, because the space that is consumed by the trails depends on the volume of data that will be processed.

Network

The following network resources must be available to support Oracle GoldenGate:

- Use the fastest network possible and install redundancies at all points of failure for optimal performance and reliability, especially in maintaining low latency on the target.

- You can configure Oracle GoldenGate Microservices Architecture to use a reverse proxy. Oracle GoldenGate MA includes a script called `ReverseProxySettings` that generates configuration file for only the NGINX reverse proxy server.
See *Reverse Proxy Support* in *Oracle GoldenGate Security Guide*.
- Configure the system to use both TCP and UDP services, including DNS. Oracle GoldenGate supports IPv4 and IPv6 and can operate in a system that supports one or both of these protocols.
- Configure the network with the host names or IP addresses of all systems that will be hosting Oracle GoldenGate processes and to which Oracle GoldenGate will be connecting.
- Oracle GoldenGate requires some unreserved and unrestricted TCP/IP network ports, the number of which depends on the number and types of processes in your configuration. See *Administering Oracle GoldenGate* for details on how to configure the Manager process to handle the required ports.
- Keep a record of the ports that you assigned to Oracle GoldenGate processes. You specify them with parameters when configuring deployments for the Microservices Architecture and for the Manager and pumps with the Classic Architecture.
- Configure your firewalls to accept connections through the Oracle GoldenGate ports.

Operating System Privileges

The following are the privileges in the operating system that are required to install Oracle GoldenGate and to run the processes:

- The person who installs Oracle GoldenGate must be granted read and write privileges on the Oracle GoldenGate software home directory.
- To install on Windows, the person who installs Oracle GoldenGate must log in as an Administrator.
- The Oracle GoldenGate Extract, Replicat, and Manager processes, and configuring deployments using the `oggca.sh` script must operate as an operating system user that has read, write, and delete privileges on files and subdirectories in the Oracle GoldenGate directory.
- For Extract processes that read from transaction logs and backups, it must operate as an operating system user that has read access to the logs and backup files.
- Oracle recommends that you dedicate the Extract and Replicat operating system users to Oracle GoldenGate. Sensitive information might be available to anyone who runs an Oracle GoldenGate process, depending on how database authentication is configured.

Security and Other Considerations

An Oracle GoldenGate Microservices deployment can be installed with various security features. When setting up a secure deployment, some information is required for proper configuration depending on whether self-signed certificates are used or provided.

Oracle GoldenGate fully supports virtual machine environments created with any virtualization software on any platform unless otherwise noted. When installing Oracle GoldenGate into a virtual machine environment, select a build that matches the database and the operating system of the virtual machine, not the host system.

**Note:**

Oracle customers with an active support contract and running supported versions of Oracle products (including Oracle GoldenGate) receive assistance from Oracle when running those products on VMware virtualized environments.

If Oracle identifies the underlying issue is not caused by Oracle's products or is being run in a computing environment not supported by Oracle, Oracle will refer customers to VMware for further assistance and Oracle will provide assistance to VMware as applicable in resolving the issue.

This support policy does not affect Oracle or VMware licensing policies.

Windows Console Character Sets

The operating system and the command console must have the same character sets. Mismatches occur on Microsoft Windows systems, where the operating system is set to one character set, but the DOS command prompt uses a different, older DOS character set. Oracle GoldenGate uses the character set of the operating system to send information to GGSCI command output; therefore a non-matching console character set causes characters not to display correctly. You can set the character set of the console before opening a GGSCI session by using the following DOS command:

```
chcp codepagenumber
```

For example, `chcp 437`.

For a code page overview, see [https://msdn.microsoft.com/en-us/library/windows/desktop/dd317752\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/dd317752(v=vs.85).aspx) and the list of code page identifiers [https://msdn.microsoft.com/en-us/library/windows/desktop/dd317756\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/dd317756(v=vs.85).aspx).

Other Operating System Requirements

The following additional features of the operating system must be available to support Oracle GoldenGate.

- To use Oracle GoldenGate user exits, install the C/C++ Compiler, which creates the programs in the required shared object or DLL.
- Gzip to decompress the Oracle GoldenGate installation files. Otherwise, you must unzip the installation on a PC by using a Windows-based product, and then FTP it to the AIX, DB2 for i, or DB2 z/OS platforms.
- For best results on DB2 platforms, apply high impact (HIPER) maintenance on a regular basis staying within one year of the current maintenance release. The HIPER process identifies defects that could affect data availability or integrity. IBM provides Program Temporary Fixes (PTF) to correct defects found in DB2 for i and DB2 z/OS.
- Oracle GoldenGate for SQL Server when installed on Linux requires the `libnsl` and `unixODBC` packages to be installed prior to launching GGSCI.
- Before installing Oracle GoldenGate on a Windows system, install the Microsoft Visual C++ 2013 Redistributable Package and the Microsoft Visual C++ 2017 Redistributable Package. These packages install runtime components of Visual C++ Libraries that are required for Oracle GoldenGate processes.

Download and install the x64 version of Visual C++ 2013 package from :

<https://support.microsoft.com/en-us/help/4032938/update-for-visual-c-2013-redistributable-package>

Download and install the x64 version of Visual C++ 2017 package from

<https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads>

- For Oracle GoldenGate for Oracle to be installed on a remote hub server, download and install the Oracle Database 19c client for the operating system platform where Oracle GoldenGate will be installed and ensure that you install the Administrator version of the client.

Prerequisites

Learn about what you need to do before installing.

Topics:

Setting TNS_ADMIN

The `TNS_ADMIN` environment variable contains the path to the TNS files.

It is recommended (but not required) to set the environment variable `TNS_ADMIN`. If this environment variable is not set, then Oracle GoldenGate looks for the `$HOME/.tnsnames.ora` or `/etc/tnsnames.ora` file. In addition, the environment variable must be set before starting the Admin Client or GGSCI. Otherwise, this variable is not detected.

If you are not using `TNS_ADMIN`, then you can use connection qualifiers such as `(DESCRIPTION=(ADDRESS=(...)))`, with TNS aliases.

A preferred technique for configuring database connections is using the EZconnect syntax. You need the username, password, hostname, port number, and service name connection information to use the EZConnect syntax.

Syntax that you need to specify in the User ID field: `username@hostname:port/service_name`

Here's an example for setting the User ID with EZConnect:

```
c##ggadmin@dc.example.com:1521/dc1.example.com
```

Specifying Oracle Variables on UNIX and Linux Systems

If there is one instance of Oracle Database on the system, then set the `ORACLE_HOME` and `ORACLE_SID` environment variables at the system level. If you cannot set them that way, then use the following `SETENV` statements in the parameter file of every Extract and Replicat group that will be connecting to the instance. The `SETENV` parameters override the system settings and allow the Oracle GoldenGate process to set the variables at the session level when it connects to the database.

```
SETENV (ORACLE_HOME = path_to_Oracle_home_location)
```

```
SETENV (ORACLE_SID = SID)
```

If there are multiple Oracle instances on the system with Extract and Replicat processes connecting to them, then you must use a `SETENV` statement in the parameter file of each process group. As input to the `SETENV` parameter, use the `ORACLE_HOME` and `ORACLE_SID` environment variables to point Oracle GoldenGate to the correct Oracle instance. For example, the following parameter file excerpts shows two Extract groups, each capturing from a different Oracle instance.

Group 1:

```
EXTRACT edbaa
SETENV (ORACLE_HOME = "/home/oracle/ora/product")
SETENV (ORACLE_SID = "oraa")
USERIDALIAS tiger1
RMTHOST sysb
RMTTRAIL /home/ggs/dirdat/rt
TABLE hr.emp;
TABLE hr.salary;
```

Group 2:

```
EXTRACT orab
SETENV (ORACLE_HOME = "/home/oracle/ora/product")
SETENV (ORACLE_SID = "orab")
USERIDALIAS tiger1
RMTHOST sysb
RMTTRAIL /home/ggs/dirdat/st
TABLE fin.sales;
TABLE fin.cust;
```

Specifying Oracle Variables on Windows Systems

If there is one instance of Oracle on the system, then the Registry settings for `ORACLE_HOME` and `ORACLE_SID` should be sufficient for Oracle GoldenGate. If those settings are incorrect in the Registry and cannot be changed, then you can set an override as follows:

1. On the Desktop or Start menu, right-click **My Computer**, and then select **Properties**.
2. In Properties, click the **Advanced** tab.
3. Click **Environment Variables**.
4. Under System Variables, click **New**.
5. For the Variable Name, enter `ORACLE_HOME`.
6. For the Variable Value, enter the path to the Oracle binaries.
7. Click **OK**.
8. Click **New** again.
9. For the Variable Name, enter `ORACLE_SID`.
10. For the Variable Value, enter the instance name.
11. Click **OK**.

If there are multiple Oracle instances on the system with Extract and Replicat processes connecting to them, then use these steps:

1. Use the preceding procedure (single Oracle instance on system) to set the `ORACLE_HOME` and `ORACLE_SID` system variables to the first Oracle instance.
2. Start all of the Oracle GoldenGate processes that will connect to that instance.
3. Edit the existing `ORACLE_HOME` and `ORACLE_SID` variables to specify the new information., then repeat the procedure for the next Oracle instance.
4. Start the Oracle GoldenGate processes that will connect to that instance.
5. Repeat the edit and startup procedure for the rest of the Oracle instances.

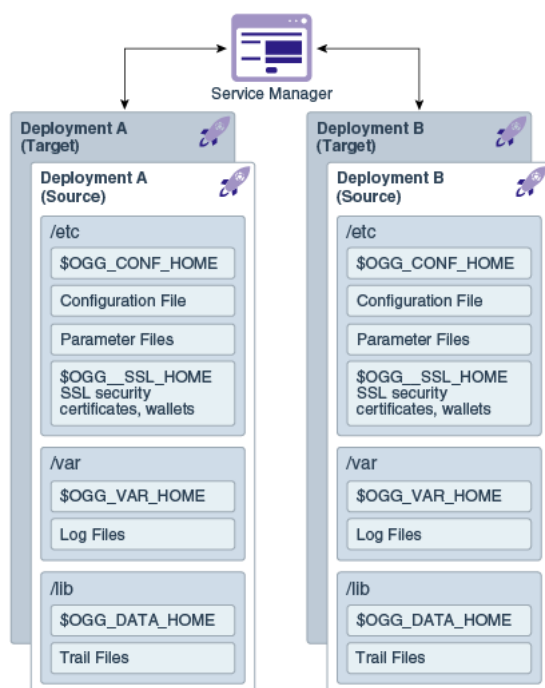
What are the Key Microservices Architecture Directories and Variables?

The Microservices Architecture is designed with a simplified installation and deployment directory structure.

This directory structure is based on the Linux Foundation Filesystem Hierarchy Standard. Additional flexibility has been added to allow parts of the deployment subdirectories to be placed at other locations in the file system or on other devices, including shared network devices. The design comprises a read-only Oracle GoldenGate home directory where Oracle GoldenGate Microservices Architecture is installed and custom deployment specific directories are created as follows:

- bin
- cfgtoollogs
- deinstall
- diagnostics
- include
- install
- inventory
- jdk
- jlib
- lib
 - instantclient
 - sql
 - utl
- OPatch
- oraInst.loc
- oui
- srvm

The following figure shows the files and directories under the Services Manager (`srvm`) directory:



The following table describes the key MA directories and the variables that are used when referring to those directories during an Oracle GoldenGate installation. When you see these variables in an example or procedure, replace the variable with the full path to the corresponding directory path in your enterprise topology.

Directory Name	Variable	Description	Default Directory Path
Oracle GoldenGate home	OGG_HOME	The Oracle GoldenGate home that is created on a host computer is the directory that you choose to install the product. This read-only directory contains binary, executable, and library files for the product.	/ogg_install_location
Deployment configuration home	OGG_CONF_HOME	The location where each deployment information and configuration artifacts are stored.	/ogg_deployment_location/etc/conf
Deployment security home	OGG_SSL_HOME	The location where each deployment security artifacts (certificates, wallets) are stored.	/ogg_deployment_location/etc/ssl
Deployment data home	OGG_DATA_HOME	The location where each deployment data artifacts (trail files) are stored.	/ogg_deployment_location/var/lib/data
Deployment variable home	OGG_VAR_HOME	The location where each deployment logging and reporting processing artifacts are stored.	/ogg_deployment_location/var

Directory Name	Variable	Description	Default Directory Path
Deployment etc home	OGG_ETC_HOME	The location where your deployment configuration files are stored including parameter files.	/ogg_deployment_location/etc

You can change the default location of all of these to customize where you want to store these files.

In a configuration where the `OGG_VAR_HOME` is a local directory and the `OGG_HOME` is a shared read-only remote directory, many deployments with local `OGG_VAR_HOME` can share one read-only shared `OGG_HOME`.

This directory design facilitates a simple manual upgrade. To upgrade, you stop the services and then set the `OGG_HOME` in the web interface (or via a REST command) and then restart the processes. On restart, Oracle GoldenGate picks up the updated environment variables. You simply switch a deployment to use a new Oracle GoldenGate release by changing the `OGG_HOME` directory path in the Service Manager to a new Oracle GoldenGate home directory, which completes the upgrade. Restart the microservices, Extract and Replicat processes.

The following table describes the programs and utilities exclusive to the MA. You should also set the `$OGG_HOME/lib/instantclient` (among other libraries that are used for the database connectivity)

Name	Description	Default Directory
adminclient	The Admin Client is a standalone command line interface used to create processes, rather than using the MA UI.	<code>\$OGG_HOME/bin</code>
adminsrvr	The Administration Service supervises, administers, manages, and monitors processes operating within an Oracle GoldenGate deployment for both active and inactive processes.	<code>\$OGG_HOME/bin</code>
distsrvr	A Distribution Service is a service that functions as a networked data distribution agent in support of conveying and processing data and commands in a distributed deployment.	<code>\$OGG_HOME/bin</code>
extract	Extract data process.	<code>\$OGG_HOME/bin</code>
oggca.sh	The MA Configuration Assistant.	<code>\$OGG_HOME/bin</code>
orapki	Utility to manage public key infrastructure elements, such as wallets and certificate revocation lists,	<code>\$OGG_HOME/bin</code>
pmsrvr	The Performance Metrics Server uses the metrics service to collect and store instance deployment performance results.	<code>\$OGG_HOME/bin</code>

Name	Description	Default Directory
recvsrvr	A Receiver Service is the central control service that handles all incoming trail files.	\$OGG_HOME/bin
replicat	Replicat data process.	\$OGG_HOME/bin
ServiceManager	A Service Manager acts as a watchdog for other services available with the MA.	\$OGG_HOME/bin
crypto		\$OGG_HOME/lib
htdocs	The MA HTML pages for all services.	\$OGG_HOME/lib
info	The various help files that support the MA HTML pages for all services.	\$OGG_HOME/lib
sql	An SQL directory that contains the healthcheck, legacy, and sharding utilities.	\$OGG_HOME/lib
SQLPLUS	The utility to run various commands.	\$OGG_HOME/lib
utl	A utility directory that contains the install, logging, reverseproxy, and sharding utilities.	\$OGG_HOME/lib

Installing Oracle GoldenGate

Learn about the steps for installing Oracle GoldenGate Microservices Architecture for the first time and includes instructions to download the base release of a new version of Oracle GoldenGate.

To download and install subsequent patches to the base release, go to the **Patches and Updates** tab of My Oracle Support at:

<https://support.oracle.com>

Also see [Installing Patches for Oracle GoldenGate Microservices Architecture](#).

Installing Oracle GoldenGate Microservices Architecture

The steps for installing Oracle GoldenGate Microservices Architecture for Oracle and Non-Oracle databases are the same. However, there are some prerequisites before you begin the installation.

Verify that you meet the operating system and required database configuration before beginning the installation. See:

- [Operating System Requirements](#)
- [Prepare Databases](#)

The Oracle GoldenGate Microservices Architecture (MA) installation involves the following steps:

1. Install the Oracle GoldenGate software. See [Performing an Interactive Installation with OUI for MA](#) and [Performing a Silent Installation with OUI](#).
2. Set the necessary environment variables for your database, if required.

 **Note:**

(Oracle only) From the Oracle GoldenGate 21c release onward, `ORACLE_HOME` and `LD_LIBRARY_PATH` do not point to any database directories. With the unified build feature, these environment variables now point to the `OGG_HOME` (sub)directories as the Oracle Database Client Software is embedded in Oracle GoldenGate.

3. Run the Oracle GoldenGate Configuration Assistant (oggca) wizard to add a deployment for the Oracle GoldenGate installation. For steps to run the OGGCA utility, see [Add a Deployment](#).

The installer registers the Oracle GoldenGate home directory (`$OGG_HOME`) with the central inventory that is associated with the selected database. The inventory stores information about all Oracle software products installed on a host if the product was installed using OUI.

Disk space is also required for the Oracle GoldenGate Bounded Recovery feature. Bounded Recovery is a component of the general Extract checkpointing facility. It caches long-running open transactions to disk at specific intervals to enable fast recovery upon a restart of Extract. At each bounded recovery interval (controlled by the `BRINTERVAL` option of the `BR` parameter) the disk required is as follows: for each transaction with cached data, the disk space required is usually 64k plus the size of the cached data rounded up to 64k. Not every long-running transaction is persisted to disk.

For complete information about Bounded Recovery, see the `BR` parameter in *Parameters and Functions Reference for Oracle GoldenGate*.

Performing an Interactive Installation with OUI for MA

Interactive installation provides a graphical user interface that prompts for the required installation information.

These instructions apply to new installations and upgrades.

1. Create a temporary staging directory into which you will install Oracle GoldenGate. For example, `mkdir /u01/stage/oggsc`.
2. Extract the installation ZIP file into the temporary staging directory. For example:

```
unzip ./fbo_ggs_Linux_x64_services.zip -d ./temp_directory
```
3. From the expanded directory, run the `fbo_ggs_Linux_x64_services_shophome/Disk1/runInstaller` program on UNIX or Linux.

The OUI Install Wizard is started.

4. On the **Select Installation Option** page, select the Oracle Database version for your environment, then click **Next**.
5. On the **Specify Installation Details** page, specify the following:
 - For **Software Location**, specify the location where Oracle GoldenGate software is to be installed. This will be your Oracle GoldenGate Home (`OGG_HOME`) after the installation is complete. If you have the `$OGG_HOME` environment variable set, this

should be the path displayed. The specified directory cannot be a registered home in the Oracle Central Inventory.

- Click **Next**.
- 6. On the **Summary** page, confirm that there is enough space for the installation and that the installation selections are correct.
 - (Optional) Click **Save Response File** to save the installation information to a response file. You can run the installer from the command line with this file as input to duplicate the results of a successful installation on other systems. You can edit this file or create a new one from a template.
 - Click **Install** to begin the installation or **Back** to go back and change any input specifications. When upgrading an existing Oracle GoldenGate installation, OUI notifies you that the software location has files or directories. Click **Yes** to continue.
 - If you created a central inventory directory, you are prompted to run the `INVENTORY_LOCATION/orainstRoot.sh` script. This script must be executed as the root operating system user. This script establishes the inventory data and creates subdirectories for each installed Oracle product (in this case, Oracle GoldenGate).

You are notified when the installation is finished.

- 7. Click **Close** to complete the installation.

Performing a Silent Installation with OUI

Silent installation from the command line interface can be performed if your system does not have an X-Windows or graphical interface or you want to perform the installation in an automated way.

Silent installations ensure that multiple users in your organization use the same installation options when installing Oracle products.

Silent installations are driven by using a response file. Response files can be saved by selecting the **Save Response File** option during an interactive Oracle Universal Installer session or by editing the `oggcore.rsp` template located in the response directory after unzipping the Oracle GoldenGate binaries.

Editing the Default Response File

- `INSTALL_OPTION` - The valid values are `ORA11g`, `ORA12c`, `ORA18c`, and `ORA19c`. Set the value based on the database version for the specific Oracle GoldenGate build to be installed.

Example:

```
INSTALL_OPTION=ORA19c
```

- `SOFTWARE_LOCATION` - Absolute path to where Oracle GoldenGate will be installed. Do not use spaces in the path and ensure that the directory has been created and is empty.

Example:

```
SOFTWARE_LOCATION==/u01/userhome/oracle/ogg19c_ora
```

- `INVENTORY_LOCATION` - Location of the Oracle Inventory files. This is optional for Windows installations.

Example:

```
INVENTORY_LOCATION=/u01/app/oraInventory
```

- `UNIX_GROUP` - The Unix group to be set for the inventory directory. Not valid for Windows installations.

Example:

```
UNIX_GROUP=oinstall
```

Installing Oracle GoldenGate

To perform a silent installation using a response file, perform the following steps:

1. Run the following command to unzip the folder that contains the Oracle GoldenGate installation program.

```
cd unzipped_directory/[fbo_]ggs_OS_database_services_shiphome/Disk1
```

2. Run the following command to launch the installer program.

```
./runInstaller -silent -nowait -responseFile absolute_path_to_response_file
```

Integrating Oracle GoldenGate Microservices Architecture into a Cluster

If you installed Oracle GoldenGate in a cluster, take the following steps to integrate Oracle GoldenGate within the cluster solution.

Oracle GoldenGate Microservices Architecture provides REST-enabled services with features including remote configuration, administration, and monitoring through HTML5 web pages, command line interfaces, and APIs.

For more information about installing and using Oracle GoldenGate in a cluster, see the [Oracle GoldenGate Microservices Architecture with Oracle Real Application Clusters Configuration Best Practices](#) technical brief.

Post-installation Tasks

Learn about any post-installation tasks that may be required after installing Oracle GoldenGate Microservices Architecture for your database.

Topics:

Software Installation Directories and Programs for Oracle GoldenGate

The following table describes the major directories of an Oracle GoldenGate Microservices installation.

Table 2-1 Directories in an Oracle GoldenGate MA installation

Directory	Description
bin	Sub-directory for most of the Oracle GoldenGate executable files.
lib	Contains libraries, utility files, and scripts.
jdk	Java Developer Kit directory
oui	Oracle Universal Installer directory

Table 2-1 (Cont.) Directories in an Oracle GoldenGate MA installation

Directory	Description
OPatch	Location of Oracle Patch Utility directory to install patches (opatch).
deinstall	Location of deinstall.sh, which is the software deinstallation script.

The following table describes the programs and utilities exclusively available with MA.

Name	Description	Default Directory
adminclient	Command line interface for Oracle GoldenGate Microservices Architecture.	\$OGG_HOME/bin
adminsrvr	The Administration Service supervises, administers, manages, and monitors processes operating within an Oracle GoldenGate deployment for both active and inactive processes.	\$OGG_HOME/bin
chkptdump	Utility to dump contents from the checkpoint files.	\$OGG_HOME/bin
distsrvr	A Distribution Service is a service that functions as a networked data distribution agent in support of conveying and processing data and commands in a distributed deployment.	\$OGG_HOME/bin
extract	Extract data process.	\$OGG_HOME/bin
logdump	Utility to open files, control the display, navigate through a file, and search, filter, view, and save data that's stored in a trail or Extract file.	\$OGG_HOME/bin
oggca.sh	The Oracle GoldenGate Microservices Configuration Assistant.	\$OGG_HOME/bin
oggerr	Retrieves a detailed explanation for an Oracle GoldenGate message.	\$OGG_HOME/bin
orapki	Utility to manage public key infrastructure elements, such as wallets and certificate revocation lists.	\$OGG_HOME/bin
pmsrvr	The Performance Metrics Service uses the metrics service to collect and store instance deployment performance results.	\$OGG_HOME/bin
recvsrvr	A Receiver Service is the central control service that handles all incoming trail files.	\$OGG_HOME/bin

Name	Description	Default Directory
replicat	Replicat data process.	\$OGG_HOME/bin
ServiceManager	A Service Manager acts as a watchdog for other microservices in Oracle GoldenGate.	\$OGG_HOME/bin
trailscan	Utility that scans transaction from trail files.	\$OGG_HOME/bin
sqlplus	An interactive tool with a command-line user interface used to connect to the Oracle Database Server.	\$OGG_HOME/lib/ instantclient
sql	An SQL directory that contains the healthcheck, legacy, and sharding utilities.	\$OGG_HOME/lib
utl	A utility directory that contains the install, logging, reverseproxy, and sharding utilities.	\$OGG_HOME/lib

Installing Patches for Oracle GoldenGate Microservices Architecture

Patching for Oracle GoldenGate refers to applying interim one-off software fixes as well as cumulative software bundle patches to an existing, lower version of the software, yet one that is in the same release label as the patch to be applied. Cumulative and one-off patches for Oracle GoldenGate can be applied on top of a base release or previously patched release, or they may be a one-off patch that should be applied to a specific Oracle GoldenGate version.

Patches for Oracle GoldenGate can be found on [My Oracle Support](#) when available, and are located under the Patches & Updates section of MOS.



Note:

When patching multiple installations that already have Deployments and a shared Service Manager configured, the Service Manager will only be patched when the Oracle GoldenGate installation where the Service Manager was first created from, gets patched.

Downloading Patches for Oracle GoldenGate

Download the appropriate patches for the Oracle GoldenGate build for each system that will be part of the Oracle GoldenGate configuration.

1. Using a browser, navigate to <https://support.oracle.com>.
2. Log in with your Oracle ID and password.
3. Select the **Patches & Updates** tab.
4. On the **Search** tab, click **Product or Family**.

5. In the **Product** field, type **Oracle GoldenGate**.
6. From the **Release** drop-down list, select the patch version that you want to download.
7. Optionally, to limit the number of patches listed in the search results, select the required platform from the **Platform** drop-down list.
8. Click **Search**.
9. In the **Patch Advanced Search Results** list, select the patch that best meets your criteria.
When you select a patch, a dialog box pops up under the build description, and then you are advanced to the patch details page.
10. Click the **Download** link for the patch and save the file to your system.

**Note:**

Before installing the patch, see [Release Notes for Oracle Database](#) for any new features, parameter changes, patching requirements, known issues, or bug fixes that affect your current configuration.

Patching Oracle GoldenGate Microservices Architecture Using OPatch

After you download the patch, set up the following prerequisites before installing the patch:

1. Download and install the most recent release of OPatch, and keep a note of the installation directory where you installed the latest release of OPatch.

Details from where to download OPatch are available at: [How To Download And Install The Latest OPatch\(6880880\) Version \(Doc ID 274526.1\)](#)

2. Download the Oracle GoldenGate patch and maintain a location for storing the contents of the patch ZIP file. This location or the absolute path is referred to as *patch_top_dir* in the subsequent steps.
3. Navigate to the *patch_top_dir* directory and run the following command to extract the contents of the patch ZIP file to the location you created previously.

```
cd patch_top_dir
unzip patch_number_version_platform.zip
```

4. Navigate to the unzipped patch directory:

```
cd patch_top_dir/patch_number_dir
```

5. Set the `ORACLE_HOME` environment variable to the Oracle GoldenGate installation directory that is to be patched:

For Linux: \$ export ORACLE_HOME=GoldenGate_Installation_Path

For Windows: > set ORACLE_HOME=GoldenGate_Installation_Path

6. Set the `PATH` environment variable to include the locations of the `ORACLE_HOME` and OPatch directories.

For Linux: \$ export PATH=\$PATH:\$ORACLE_HOME:/OPatch

For Windows: >set PATH=%PATH%;%ORACLE_HOME%;C:\OPatch

7. Verify the Oracle inventory, which OPatch accesses to install the patches. To verify the inventory, run the following command:

```
opatch lsinventory
```

If the command displays any errors, contact Oracle Support to resolve the issue.

8. Run the OPatch prerequisites check and verify that it passes.

```
opatch prereq CheckConflictAgainstOHWithDetail -ph ./
```

If any errors are displayed, identify the error type. OPatch categorizes conflicts in the following types:

- Conflicts with a patch already applied to the `ORACLE_HOME`: In this case, stop the patch installation and contact Oracle Support Services.
 - Conflicts with a patch already applied to the `ORACLE_HOME` that is a subset of the patch you are trying to apply: In this case, continue with the patch installation because the new patch contains all the fixes from the existing patch in the `ORACLE_HOME`. The subset patch will automatically be rolled back prior to the installation of the new patch.
9. Before patching Oracle GoldenGate, if you have any deployments for the installation, ensure that you shut down all processes such as Extracts, Replicats, and Distribution paths, and stop all services for the deployments.

This can be done in the Administration Service's and Service Manager's WebUI, or in the Admin Client.

If using the Admin Client, perform the following steps to connect to each deployment and stop all processes.

10. If using the Admin Client, connect to each deployment and stop all processes.

- a. Start the Admin Client and connect to the deployment.

```
/GoldenGate_Installation_Path/bin/adminclient
OGG (not connected) 1>CONNECT https://host:srv_mgrport
DEPLOYMENT <deployment-name> AS <user> PASSWORD <password>
```

- b. Stop the Extract and Replicat processes and the Distribution Paths.

```
STOP ER *
STOP DISTPATH ALL
```

- c. Stop the services for the deployment and verify that they are all stopped:

```
STOP SERVICE *
STATUS SERVICE *
```

- d. Exit the Admin Client and stop the Service Manager:

```
OGG (https://host:port deployment-name) exit
##Command for Service Manager not registered as a service/daemon
export OGG_VAR_HOME=OGG_SRMGR_DIRECTORY/var
export OGG_ETC_HOME=OGG_SRMGR_DIRECTORY/etc
OGG_SRMGR_DIRECTORY/bin/stopSM.sh
```

```
##Command for Service Manager registered as a service/daemon
```

For Linux: \$ sudo systemctl stop OracleGoldenGate

For Windows: To stop the Service Manager for Windows, use the Windows Services applet (services.msc) and stop the Oracle GoldenGate Service Manager service.

11. Disconnect all user sessions to the deployment as well as close all running Oracle GoldenGate programs, including Admin Client.

Perform the following steps to install the patch:

12. Install the patch by running the following command:

```
opatch apply
```

When the `OPatch` command starts, it validates the patch and ensures that there are no conflicts with the software already installed in `ORACLE_HOME` of the Oracle GoldenGate release.

13. After the patch installation completes, run the following command to verify that the Oracle inventory contains the installed patch:

```
opatch lsinventory
```

Note:

For Oracle GoldenGate for PostgreSQL installations patched to release version 21.8.0.0.2 and later, prior to restarting the Extracts and Replicats, update the DSN entries in the `odbc.ini` file to take advantage of the new driver version.

14. After the patch installation completes, start the Service Manager, the services, and Oracle GoldenGate processes.

- a. Start the Service Manager:

For Linux:

```
##Command for Service Manager not registered as a service/daemon
```

```
$ export OGG_VAR_HOME=OGG_SRVMGR_DIRECTORY/var
```

```
$ export OGG_ETC_HOME=OGG_SRVMGR_DIRECTORY/etc
```

```
$ OGG_SRVMGR_DIRECTORY/bin/startSM.sh
```

```
##Command for Service Manager registered as a service/daemon
```

```
$ sudo systemctl start OracleGoldenGate
```

For Windows: Use the Windows Services applet (services.msc) and start the Oracle GoldenGate Service Manager service.

- b. Start the Admin Client and connect to the deployment.

```
/GoldenGate_Installation_Path/bin/adminclient
```

```
OGG (not connected) 1>CONNECT https://host:srvmgr_port DEPLOYMENT
deployment-name AS user PASSWORD password
```

- c. Start services for the deployment and verify that they are all running:

```
START SERVICE *  
STATUS SERVICE *
```

- d. Start the Extract, Replicat and Distribution paths:

```
START ER *  
START DISTPATH ALL
```

Uninstalling the Patch for Oracle and Non-Oracle Databases Using OPatch

To uninstall the patch, follow these steps:

1. Install the latest OPatch version, set the required environment variables, and stop the Oracle GoldenGate processes and services. The patch installation steps are documented in the previous topic.
2. Navigate to the *patch_top_dir/patch_number* directory:

```
$ cd patch_top_dir/patch_number
```

3. Uninstall the patch by running the following command:

```
$ opatch rollback -id patch_number
```

4. Start the services from the Oracle GoldenGate home.

Uninstalling Oracle GoldenGate Microservices Architecture

Learn about uninstalling Oracle GoldenGate Microservices Architecture processes and files from the host in Linux, UNIX, and Windows environments.

It is assumed that you no longer need the data in the Oracle GoldenGate trails, and that you no longer need to preserve the current Oracle GoldenGate environment. To preserve your current environment and data, make a backup of the Oracle GoldenGate directory and all subdirectories before starting this procedure.

Before uninstalling Oracle GoldenGate Microservices Architecture, you must stop the Service Manager and all the deployments.

Removing Deployments and Service Manager

Learn how to remove a deployment using OGGCA.

Removing Deployments and Service Manager Using Oracle GoldenGate Configuration Assistant

To remove a deployment using Oracle GoldenGate Configuration Assistant (OGGCA), perform the following steps:

1. Connect to the Administration Server of all deployments to be removed, and stop any running Extracts and Replicats.

2. Perform the following step for Linux and Windows systems:
 - In Linux systems, run the command `./oggca.sh` from the `$OGG_HOME/bin` directory to launch the Oracle GoldenGate Configuration Assistant (OGGCA).
 - In Windows systems, right-click the `oggca.bat` file and select **Run as administrator**. This file is located in the `OGG_HOME\bin` directory.
3. Select the **Existing Service Manager** option and click **Next**.
4. Select **Remove Existing Oracle GoldenGate** deployment and click **Next**.
5. Follow the steps in the OGGCA wizard to remove the deployment.
6. Repeat the steps to remove multiple deployments and the Service Manager.


Using Oracle GoldenGate Configuration Assistant - Silent

To run the Configuration Assistant in silent mode, execute it with the `-silent -responseFile fullPathToResponseFile` flags.

The properties expected to be set in the response file for removing a deployment are:

```
CONFIGURATION_OPTION,
DEPLOYMENT_NAME,
ADMINISTRATOR_USER,
ADMINISTRATOR_PASSWORD,
HOST_SERVICEMANAGER,
PORT_SERVICEMANAGER,
SECURITY_ENABLED,
REMOVE_DEPLOYMENT_FROM_DISK
```

Files to be Removed Manually

Operating System	Files to be Removed Manually to Unregister an Existing Service Manager
Linux 6	<ul style="list-style-type: none"> • <code>/etc/init.d/OracleGoldenGate</code> • <code>/etc/rc.d/*OracleGoldenGate</code> • <code>/etc/rc*.d/*OracleGoldenGate</code> • <code>/etc/oggInst.loc</code>
<div>  Note: Linux 6 is not certified for Oracle GoldenGate 21c (21.3.0). This information may be required when trying to perform upgrades or downgrades. </div>	
Linux 7 and Linux 8	<code>/etc/systemd/system/OracleGoldenGate.service</code>

Uninstalling Microservices Architecture with Oracle Universal Installer



Note:

It's important to remove all deployments prior to uninstalling Oracle GoldenGate home directory.

To uninstall Oracle GoldenGate Microservices Architecture with Oracle Universal Installer:

1. Navigate to the following directory:

```
/$OGG_HOME/deinstall/
```

2. Run the command:

On UNIX and Linux: `./deinstall.sh`

On Windows: `\deinstall.bat`

See [Files to be Removed Manually](#) for steps that you may need to perform manually.

Silent Uninstallation

If you want perform a silent uninstallation, use the command:

```
deinstall.sh -silent
```

Make sure that you've set the `OGG_HOME` variable correctly as the uninstallation is silent so you will not be prompted.

See the following example for a silent uninstallation:

```
OS> ./deinstall.sh
ALERT: Ensure all the processes running from the current Oracle Home are
shutdown prior to running this software uninstallation script.Proceed with
removing Oracle GoldenGate home:
/net/xyz02/scratch/scott/view_storage/scott_x19200x/local/ggtest/
install_200714
      (yes/no)? [no] yes
Starting Oracle Universal Installer...
Checking swap space: must be greater than 500 MB.
Actual 11648 MB
PassedPreparing to launch Oracle Universal Installer from /tmp/
OraInstall2020-08-19_10-52-30AM.
      Please wait ...
Oracle Universal Installer, Version 12.2.0.1.4 ProductionCopyright (C) 1999,
2016, Oracle. All rights reserved.Starting deinstallDeinstall in progress
(Wednesday, August 19, 2020 10:52:33 AM
PDT)..... 100%
Done.Deinstall successful
      OS> ./deinstall.sh -silentALERT:
Ensure all the processes running from the current Oracle Home are shutdown
prior to running this software uninstallation script. Starting Oracle
Universal Installer... Checking swap space: must be greater than 500 MB.
Actual 11647 MB
      Passed Preparing to launch Oracle Universal Installer from /tmp/
```

```
OraInstall2020-08-19_10-43-25AM.  
Please wait ...  
Oracle Universal Installer, Version 12.2.0.1.4 Production Copyright (C) 1999,  
2016, Oracle. All rights reserved.  
Starting deinstall  
Deinstall in progress (Wednesday, August 19, 2020 10:43:29 AM PDT)  
..... 100% Done.  
Deinstall successful
```

3

Deploy

Learn about the OGGCA utility and accessing the Service Manager and Deployment configurations for the first time, using the login credentials for Oracle GoldenGate.

Deployments are created after Oracle GoldenGate software is installed. The Oracle GoldenGate Configuration Assistant (OGGCA) utility is used to create deployments and the Service Manager process on a host machine.

The OGGCA utility has many functions, which can be performed by running this program from the /bin folder of the Oracle GoldenGate software installation directory (**\$OGG_HOME/bin**). You can use OGGCA to perform the following tasks:

- Add the Service Manager to a host machine after completing the Oracle GoldenGate installation.
- Add or remove deployments from a Service Manager.
- Create users for accessing the Service Manager and user deployments and enable a strong password policy.
- Integrate with XAG when using Oracle GoldenGate with Oracle Grid Infrastructure.
- Save the OGGCA response file that contains the configuration details of the Service Manager and deployment.
- Configure environment variables.
- Enable security and upload client, service, and trusted root CA certificates for the Service Manager and deployment.
- Enable the Configuration Service to store configuration data to a specified filesystem or Oracle database server.
- Enable and configure the StatsD server to send performance data.

Add a Deployment

Follow the instructions on this page to add a deployment using the OGGCA wizard.

Using OGGCA Wizard for Deployment

This section discusses using the OGGCA wizard for deployment.

Start the OGGCA Wizard

Adding deployments is the first task in the process of setting up a data replication platform. Deployments are managed from the Service Manager. After completing the Oracle GoldenGate MA installation, you can add initial and subsequent deployments using the Oracle GoldenGate Configuration Assistant (OGGCA) wizard.

**Note:**

Oracle recommends that you maintain a single Service Manager per host, to avoid redundant upgrade and maintenance tasks with Oracle GoldenGate releases.

To start the OGGCA wizard:

1. Navigate to the `$OGG_HOME/bin` directory to access the Oracle GoldenGate Configuration Assistant (`oggca`) utility.
2. Run the `oggca.sh` program on UNIX or `oggca.bat` on Windows.

The Oracle GoldenGate Configuration Assistant (`oggca`) wizard is displayed.

The following topics provide details on the configuration that you can set on each of the OGGCA screens.

Select Service Manager Options

1. Select the **Create a New Service Manager** option if you are running OGGCA for the *first time*. When you run OGGCA for the first time, the **Existing Service Manager** option is disabled. If it's not the first time, then you can choose the **Existing Service Manager** option, which would load the Service Manager port and other settings as configured for the existing Service Manager. The deployment would be added to this Service Manager. In most configurations, there is only one Service Manager to manage multiple deployments.
2. For a new Service Manager, browse and enter the directory that you want to use for your deployment in the **Service Manager Deployment Home** text box. Oracle recommends that you create a `ServiceManager` directory within the deployment sub-directory structure to store the Service Manager files.
3. Enter the connection details for the Service Manager:
 - a. **Listening hostname/address:** Enter a hostname such as `localhost` or the IP address of the server where Service Manager will run.
 - b. **Listening Port:** Enter a unique port number that the Service Manager will listen on, or choose the port already in use if selecting an existing Service Manager.
4. (Optional) Select the option **Register the Service Manager as a system service (daemon)** to avoid manually starting and stopping it if the machine is rebooted. If there is an existing Service Manager registered as a service and you select a new Service Manager to register as a service, an alert is displayed indicating that you cannot register the new one as a system service. All other Service Managers are started and stopped using scripts installed in the `bin` directory of the deployment.

You cannot register an existing Service Manager as a system service. Enter a unique port number that the Service Manager will listen on, or choose the port already in use, if selecting an existing Service Manager.

5. (Optional) Select the **Integrate with XAG** option to integrate your deployment with an Oracle Grid Infrastructure for Oracle Database. This is only available for Oracle database in a cluster environment. This option cannot be used when running your Service Manager as a system service.
6. Click **Next**.

Configuration Options

In the **Configuration Options** step, you can add or remove deployments.

You can only add or remove one deployment for one Service Manager at a time.

**Note:**

Ensure that your Service Manager is up and running prior to launching OGGCA.

Deployment Details

Deployment Details

1. Enter the deployment name using these conventions:
 - Must begin with a letter.
 - Can be a standard ASCII alphanumeric string not exceeding 32 characters.
 - Cannot include extended ASCII characters.
 - Special characters that are allowed include underscore ('_'), forward slash ('/'), dash ('-'), period ('.').
 - Cannot be "ServiceManager".
2. Select the **Enable FIPS** check box to enable Oracle GoldenGate services to use FIPS-compliant libraries.
3. (Oracle Database only) Select **Enable Sharding** to use the database sharding feature in your deployment. The schema must be `ggadmin`.
4. Enter or select the Oracle GoldenGate installation directory. If you have set the `$OGG_HOME` environment variable, the directory is automatically populated. Otherwise, the parent directory of the `oggca.sh` (Linux) or `oggca.bat` (Windows) script is used.
5. Click **Next**.

Select Deployment Directories

Select Deployment Directories

1. Enter or select a deployment directory where you want to store the deployment registry and configuration files. When you enter the deployment directory name, it is created if it doesn't exist. Oracle recommends that you do *not* locate your deployment directory inside your `$OGG_HOME` and that you create a separate directory for easier upgrades. The additional fields are automatically populated based on the specified deployment directory.

 **Note:**

The deployment directory name (user deployment directory) needs to be different than the directory name chosen in the first screen (Service Manager deployment directory).

2. You can customize the deployment directories so that they are named and located differently from the default.
3. Enter or select different directories for the various deployment elements.
4. Click **Next**.

Specify Environment Variables

Environment Variables

Enter the requested values for the environment variables. Double-click in the field to edit it. You can copy and paste values in the environment variable fields. Make sure that you tab or click outside of the field after entering each value, otherwise it's not saved. If you have set any of these environment variables, the directory is automatically populated.

OGG_HOME

The directory where you installed Oracle GoldenGate. This variable is fixed and cannot be changed.

 **Note:**

On a Windows platform, ensure that there's no space in the `OGG_HOME` directory path otherwise OGGCA will not run.

IBMCLIDRIVER

Valid for DB2 z/OS.

Specifies the location where the IBM Data Server Driver for ODBC and CLI (IBMCLIDRIVER) software is installed.

LD_LIBRARY_PATH

This variable is used to specify the path to search for libraries on UNIX and Linux. It may have a different name on some operating systems, such as `LIBPATH` on IBM AIX on POWER Systems (64-Bit), and `SHLIB_PATH` on HP-UX. This path points to the Oracle GoldenGate installation directory and the underlying instant client directory by default. If you are using User Exits, then append the `LD_LIBRARY_PATH` variable with the path to the additional shared libraries of the User Exit.

TNS_ADMIN

Valid for Oracle database.

This variable is recommended and points to the directory location containing `tnsnames.ora`, which has the database connection details. If this variable is not set, Oracle GoldenGate looks for `$HOME/.tnsnames.ora` or `/etc/tnsnames.ora`. You need to create the `tnsnames.ora` file

with the connection data and place it in the \$OGG_HOME/etc. Here's a sample structure of the file:

```
# tnsnames.ora Network Configuration File:
# Generated by Oracle configuration tools.

LISTENER_ORCL19 =
  (ADDRESS = (PROTOCOL = TCP) (HOST = eastdb.us.oracle.com) (PORT = 1521))

ORCL =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP) (HOST = eastdb.us.oracle.com) (PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = orcl.us.oracle.com)
    )
  )
```

For example: TNS_ADMIN=/u01/app/oracle/network/admin

STREAMS_POOL_SIZE

For Oracle Database Sharding only. This variable is mandatory for sharded databases. Use the default or set your pool size value that is at least 1200MB.

TZ

Valid for MySQL.

Use the following query to determine the timezeone of the MySQL database:

```
SELECT @@global.time_zone;mysql> select
@@global.time_zone;+-----+| @@global.time_zone |
+-----+| SYSTEM |+-----+1 row in set (0.00 sec)2
```

If it returns the timezone as `SYSTEM`, then it indicates that the database timezone is the same as the system timezone.

To know the system timezone, you can run the following query on your MySQL database:

```
mysql> select @@system_time_zone;+-----+| @@system_time_zone |
+-----+| UTC |+-----+1 row in set (0.00 sec)3
```

Make sure that the database timezone and the timezone of the system where the Oracle GoldenGate instance is running are the same.

Alternatively, you can set the `TZ` variable for the deployment to the same value as the database timezeone. Use the following command to check the `TZ` variable on the shell where you are going to invoke `TZ`:

```
linux# echo $TZa
```

Use the following command to set `TZ` on the shell where you start your Oracle GoldenGate command line (Admin Client or GGSCI) from:

```
linux# export TZ UTC
```

ODBCINI

Valid for Oracle GoldenGate installed on Linux for PostgreSQL databases. Specifies the full path of the ODBC file used to store Data Source Names (DSN) for connectivity to a PostgreSQL database. For example, `ODBCINI=/etc/odbc.ini`

JAVA_HOME

If this variable is present during deployment creation, it will automatically be populated, otherwise you can set it to be:

```
export JAVA_HOME=$OGG_HOME/jdk
```

You can add additional environment variables to customize your deployment or remove variables.

Click **Next**.

Service Manager Administrator Account

Administrator Account

To choose between Identity Cloud Service (IDCS) or local credential setup, define your Service Manager administrator user.



Note:

The option to set up IDCS-enabled administrator account is not applicable when you run OGGCA for the first time. Only after creating and enabling the Authorization Profile, you can set up the Administrator Account for accessing IDCS. See [Enabling Authorization Profile](#).

1. Enter a user name and password that you want to use to sign in to the Oracle GoldenGate MA Service Manager and the other services. This user is the security user for this deployment.

If you are adding a deployment to an existing Service Manager and intend to use IDCS (as your external Identity Provider) for user authentication, then specify the user credentials for the IDCS server. As a prerequisite to providing the credentials for accessing the IDCS server, you need to enable the Authorization Profile from the Service Manager deployment.



Note:

For Administrator Account, you must enter a user and password for a provisioned external IDP identity that is mapped to the SECURITY group previously configured for the Service Manager deployment.

Select the **Enable strong password policy in the new deployment** checkbox to ensure setting a highly secure password for your user account. This password policy applies for

your localCredentialStore only but not for IDCS default settings. See Manage Oracle Identity Cloud Service Password Policies in *Administering Oracle Identity Cloud Service* guide.

The strong password policy for **localCredentialStore** has the following requirements:

- At least one lowercase character [a...z]
- At least one uppercase character [A...Z]
- At least one digit [0...9]
- At least one special character [- ! @ % & * . #]
- The length should be between 8 and 30 characters.

For details on the different types of users, see [How to Add Users](#). If you are using an existing Service Manager, you must enter the same log in credentials that were used when adding the first deployment.

2. Select the check box that allows you to enable a strong password policy for your new deployment. If you select this option, then the password must adhere to restrictions, otherwise an error occurs, which requires you to specify a stronger password.
3. Click **Next**.

Local Administrator Account Credentials

On this screen, enter the user credentials for the local administrator for the new deployment. If you want to enable IDCS for this new deployment, you can do so by enabling the authorization profile.



Note:

If Service Manager is enabled for IDCS, it can continue to manage the new deployment, which uses local administrator credentials, even if the new deployment is not enabled for IDCS.

Specify Security Options

Security Options

1. You can choose whether or not you want to secure your deployment. Oracle recommends that you enable SSL/TLS security.
If you do not want to use security option on the source endpoint, deselect the check box.
2. When you deselect the SSL/TLS check box, the option **This non-secure deployment will be used to send trail data to a secure deployment** stays enabled. Select this check box to set up a secure target deployment to communicate with a non-secure source deployment. In this case, certificates are required for the client only.
However, you must enable security if configuring for Oracle GoldenGate sharding support for Oracle Database.
3. For the Server (wallet or certificate), select the option to use a Wallet or Certificate. Provide the location of the wallet directory and if you are using an existing wallet, it must have the appropriate certificates already imported into it. If you choose to use a certificate, enter the corresponding pass phrase.

When using a self-signed certificate, a new Oracle Wallet is created in the new deployment and these certificates are imported into it. For certificates, enter the location of the private key file and the pass phrase. The private key files must be in the PKCS#8 format.

4. For the Client side, select either the wallet directory or certificate. Provide the wallet directory on the client side or the certificate details for the client. If you select the **This non-secure deployment will be used to send trail data to a secure deployment**, then you only need to specify the client side details (wallet or certificate of the target deployment). This option is useful when the Distribution Service from the source deployment is unsecured whereas the Receiver Service on the target deployment is secured. So, the sender may be configured for public access while the Receiver Service requires authentication and authorization, which is established using PKI before the incoming data is applied.

For more information, see [Creating a Self-Signed Root Certificate](#) .

Also see: [Add a Target-Initiated Distribution Path](#).

5. Click **Next**.

Advanced Security Settings

If security is enabled, then this screen is displayed with the encryption options TLS 1.1 and TLS 1.2. **TLS 1.2** is selected by default. When you open the **Advanced Security Settings** for the first time with TLS 1.2, the available cipher suites are listed.

1. Use the arrows to add or remove cipher suites.
2. Use Up and Down to reorder how the cipher suites are applied and click **Next**.

Advanced Security Settings

(If Security is enabled) On the page, the encryption options TLS 1.1 and TLS 1.2 are available. TLS 1.2 is selected by default.

When you open the Advanced Security Settings for the first time with TLS 1.2, the following cipher suites are listed:

```
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_128_GCM_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
```

```
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256  
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384  
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384
```

1. Use the arrows to add or remove cipher suites or the **Up** and **Down** to reorder how the cipher suites will be applied.
2. Click **Next**.

Sharding Options

Sharding Options

If Sharding was enabled in the previous step, then you can configure the sharding options on this screen.

1. Locate and import your Oracle GoldenGate Sharding Certificate. Enter the distinguished name from the certificate that will be used by the database sharding code to identify itself when making REST API calls to the Oracle GoldenGate MA services.
2. Enter a unique name for the certificate.
3. Click **Next**.

Port Settings

Port Settings

1. Enter the Administration Server port number, and then when you leave the field the other port numbers are populated in ascending numbers. Optionally, you can enter unique ports for each of the services.
2. Select **Enable Monitoring** to use the Performance Metrics Server.
3. Click inside the Performance Metrics Server port fields to populate or enter the ports you want to use. Ensure that you choose available ports for TCP. See [Protocols for Performance Monitoring on Different Operating Systems](#).

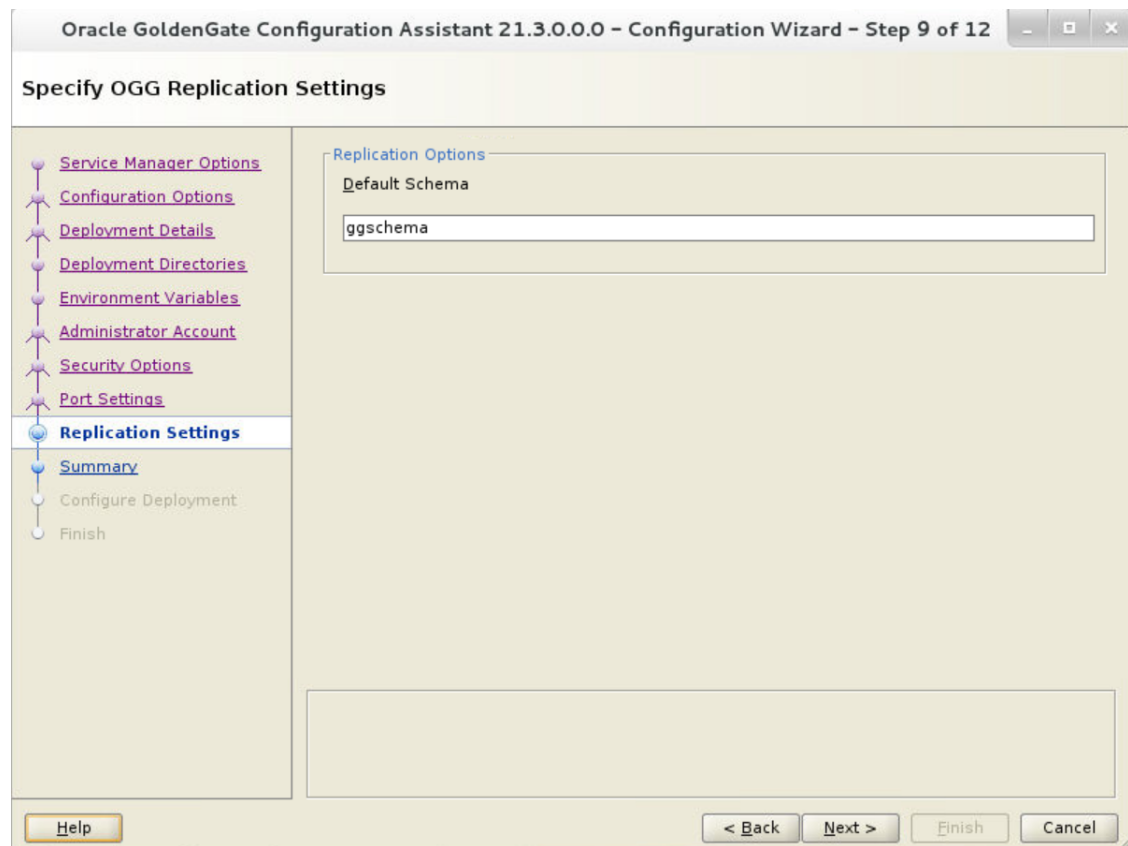
You can change the TCP port from the Service Manager console after the deployment is done. For more information on PMSRVR, see [ENABLEMONITORING](#).

4. Select the type of datastore that you want the Performance Metrics Server to use, the default Berkeley Database (BDB) data store or Open LDAP Lightning Memory-Mapped Database (LMDB). You can also designate the Performance Monitor as a Critical Service if integrating the Service Manager with XAG.

For BDB information, see [Oracle Berkeley DB 12c Release 1](#). For LMDB information, see <http://www.lmdb.tech/doc/>.

5. Select the location of your datastore. BDB and LMDB are in-memory and disk-resident databases. The Performance Metrics Server uses the datastore to store all performance metrics information.
6. Click **Next**.

Replication Settings



1. Enter the Oracle GoldenGate default schema that you want to use to store the replication objects such as checkpoint and heartbeat tables.

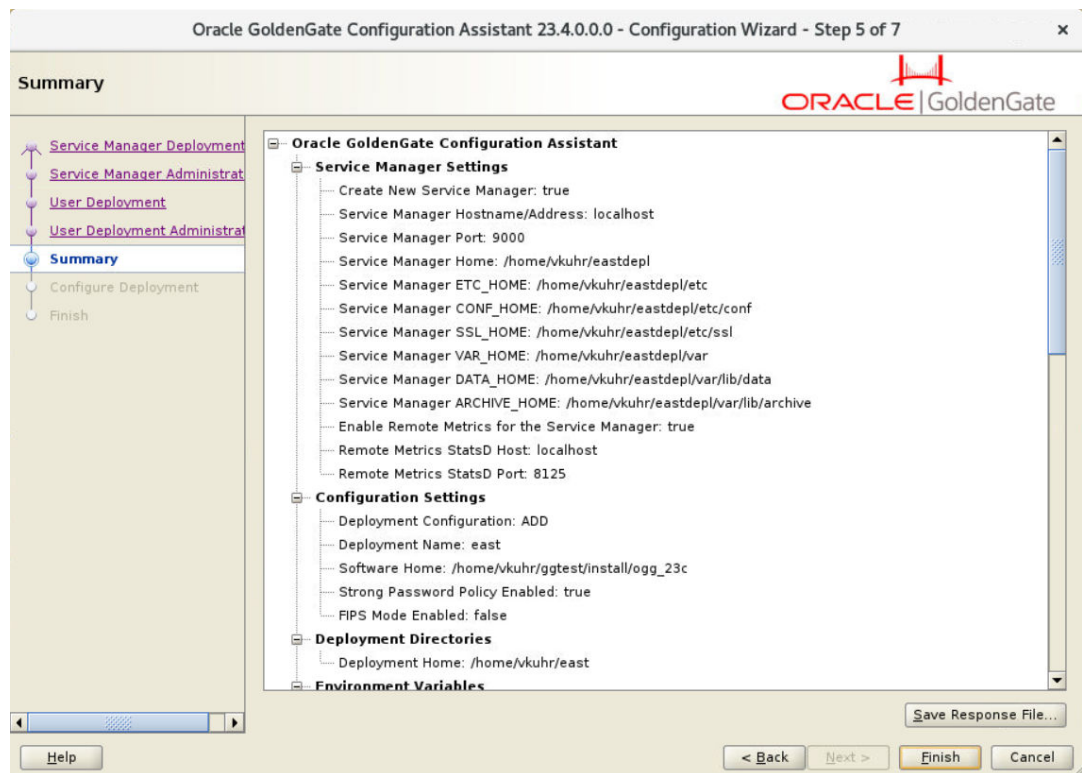
 **Note:**

OGGCA doesn't connect to the database, so it cannot validate the schema. The schema specified in OGGCA is written to the GLOBALS file as a default schema. When creating an Extract, if you do not specify a replication schema, Extract will use this schema.

2. Click **Next**.

Summary

1. Review the detailed configuration settings of the deployment before you continue, as shown in the following image.



- (Optional) You can save the configuration information to a response file. Oracle recommends that you save the response file. You can run the installer from the command line using this file as an input to duplicate the results of a successful configuration on other systems. You can edit this file or a new one from the provided template.

Note:

When saving to a response file, the administrator password is not saved for security reasons. You must edit the response file and enter the password if you want to reuse the response file for use on other systems.

- Click **Finish** and then click **Next**.

Configure Deployment

This screen displays the progress of the deployment creation and configuration. There could be some notifications during the progress if the Service Manager is registered as a service.

A pop-up appears that directs you how to run the script to register the service. The Configuration Assistant verifies that these scripts have been run. If you did not run them, you are queried if you want to continue. When you click **Yes**, the configuration completes successfully. When you click **No**, a temporary failed status is set and you click **Retry** to run the scripts.

Click **Ok** after you run the script to continue.

After the creation and configuration process completes, you'll see a message that the deployment is added successfully. Click **Next**.

Finish

On the Finish screen, click **Close** to exist OGGCA.

Add a Deployment to an Existing Service Manager

To add a deployment to an existing Service Manager, run the OGGCA utility using the following commands:

```
cd $OGG_HOME/bin
./oggca.sh
```

Use the following steps to add a deployment to the existing Service Manager:

1. The OGGCA configuration wizard is displayed. The **Existing Service Manager** option is preselected in the Service Manager Deployment screen and the Service Manager Connection details are displayed. Click **Next**.
2. On the Select Configuration Options screen, the **Add new GoldenGate deployment** option is preselected. Click **Next**.
3. On the Specify Deployment Details screen, enter a name of the deployment, enable Sharding (if required), and specify the home directory where Oracle GoldenGate is installed. Click **Next**.
4. On the Specify Deployment Directories screen, enter a deployment directory where you want to store the deployment registry and configuration files. When you enter the deployment directory name, it is created if it doesn't exist. Oracle recommends that you do *not* locate your deployment directory inside your `$OGG_HOME` and that you create a separate directory for easier upgrades. The additional fields are automatically populated based on the specified deployment directory.

 **Note:**

The deployment directory name (user deployment directory) needs to be different than the directory name chosen in the first screen (Service Manager deployment directory).

You can customize the deployment directories so that they are named and located differently from the default.

5. Specify different directories for the various deployment components and click **Next**.
6. Specify the environment variables in the same way as you would do for a new deployment. See [Specify Environment Variables](#).
7. Specify the Administrator Account details, which would be used to log in to the deployment. See [Service Manager Administrator Account](#).
8. On the Specify Security Options screen, select the options to use SSL/TLS and the client, server certificate details. See [Specify Security Options](#) and [Advanced Security Settings](#).
9. On the Specify Port Settings screen, provide the port numbers for the Microservices. Also select the Performance Monitoring check box if you want to monitor process performance for Oracle GoldenGate processes. See [Port Settings](#).

10. On the Specify OGG Replication Settings screen, enter the name of the replication schema to be used with the deployment.
11. Review the settings on the Summary screen and click Next. You can save the response file at this stage.
12. After the deployment is added successfully, click Finish to exit from the OGGCA wizard.

Add a Deployment in Silent Mode using OGGCA

To add a deployment in the silent mode, perform the following steps:

- Open the Deployment response file template with extension .rsp, available at this location in Oracle GoldenGate:
`ogg-home/inventory/response`
- Follow the instructions specified in the file to edit and then save the file with a different name, such as `oggca.rsp`.
- Create the Deployment by running the following command:

```
ogg-home/bin/oggca.sh -silent -responseFile path/oggca.rsp
```

Example:

```
/u01/app/ogg/bin/oggca.sh -silent -responseFile /u01/app/ogg/inventory/response/oggca.rsp
```

First Access to the Deployment from the Service Manager

To start using your Oracle GoldenGate Microservices deployment, you have to connect to the Service Manager:



Note:

When you log into the Service Manager for the first time, it is recommended to change the password.

1. Open a web browser and connect to the Service Manager that you created with Oracle GoldenGate Configuration Assistant. The URL is similar to `http://localhost:9001`, where 9001 is the port where you have deployed your Service Manager instance. For a secure deployment, the URL is similar to `https://localhost:9001`.
2. Enter the user name and password you created during deployment and sign in.

In the Service Manager, you can check if the Service Manager and all the other microservices are up and running. Use the links to connect you to their specific interfaces, review details, and administer your deployments.

For more information on setting up the Service Manager as a daemon service, see [Select Service Manager Options](#)

Add Deployment Users from the Service Manager

Each deployment has its own set of users with specific roles. The administrator account user, which is created when the Service Manager is created for a host using OGGCA, can log into the Service Manager and other microservices. This user can also create users with specific roles to access or operate Oracle GoldenGate processes. This administrator account user can access all deployments that are added to this existing Service Manager.

However, all subsequent users created from either the Service Manager or Administration Service are associated with the specific deployment. These users are not available with other deployments on the same host server.

The other users are specific to the MA deployment and the security user needs to create users to every MA deployment individually.

You can create users from the Service Manager or the Administration Service. See [Add Deployment Users from the Administration Server](#) for steps to create users.

For Oracle database, see Granting the Appropriate User Privileges to learn about specifying database privileges for Oracle GoldenGate.

For non-Oracle databases, see the user privileges section for DB2 z/OS, MySQL, PostgreSQL, SQL Server.

You can create users for that deployment by performing the following steps:

1. Log in to the Administration Service.
2. From the left navigation pane, select User Administration.
3. Click Users (+) to add users.
4. Enter the following details for the user:
 - **Authenticated By:** User authentication can be done with a user ID and password method or by using certificates. Select the type of authentication for the user from the drop down.
 - **Role:** User roles include Administrator, Security, User, and Operator. Select the user role based on the functions that the user needs to be perform. The following table describes these user roles:

Role ID	Privilege Level
User	Allows information-only service requests, which do not alter or effect the operation of either the MA. Examples of Query/Read-Only information include performance metric information and resource status and monitoring information.
Operator	Allows users to perform only operational actions, such as creating, starting and stopping resources. Operators cannot alter the operational parameters or profiles of the MA service.
Administrator	Grants full access to the user, including the ability to alter general, non-security related operational parameters and profiles of the service.
Security	Grants administration of security related objects and invoke security related service requests. This role has full privileges.

- If you selected the **Password** option from the **Authenticated By** drop down, then specify the user ID and password for the Oracle GoldenGate user.
- If you selected the **Certificate** option from the **Authenticated By** drop down, then click **Upload** to upload the related Certificate or paste it in the text box. This certificate is validated for user authentication when connecting remote deployments. This type of user authenticates itself by presenting a client certificate to the target deployment to allow connectivity. The common name (in the certificate that will be presented such as **CN="certuser"**) is used when setting up the DISTPATH (target authentication method) to connect different source and target deployments.

 **Note:**

The certificate is associated with the user and not saved by the Oracle GoldenGate service. When presented for authentication, the Oracle GoldenGate deployment service first authenticates that the certificate presented can be trusted and then checks to see that the common name in the certificate has been registered as a valid user. If yes, it will assign the appropriate user role.

If the user needs to set up a trusted CA certificate, then in the CA Certificates section, you can click **Enter** and paste the CA certificate in the text box. You can also click **Upload** to upload the CA certificate file.

5. Click **Submit**. The new user shows in the list of Users in the Users table.
6. You can also edit or delete a user from the Action column of the Users table.

You can switch the User Type from Basic to Certificate or the other way around. You can also change the password for the user, if required.

Click **Submit** to confirm the modifications to the user attributes.

Users cannot be changed. You must delete a user, and then add it again. However, you can modify or edit a user's attributes, by clicking the **Edit User** (pencil) in the Action column of the **Users** table.

You can switch the authenticated by option from **Password** to **Certificate** or the other way around.

You can also change the password for the user, if required.

Click **Submit** to confirm the modifications to the user attributes.

Add Deployment Users from the Administration Server

Oracle GoldenGate MAUsers can be created from the Administration Server, once you log in using the credentials created at the time of configuring the deployment.

This is an optional step with which you can easily identify if replication (setup) is working or not. To create a user, perform the following tasks:

1. Click **Administrator** from the left navigation pane of the Administration Server.
2. Click **+** to add a user.
3. Enter the required credentials in the fields.

4. Make sure that you select a role from the **Role** drop-down list. The available roles are: Administrator, Security, User, and Operator.
5. Click **Submit**.
The new user is listed in the Users table including the role and information that you supplied.

Manage Deployments from the Service Manager

The ServiceManager can be configured in three different modes:

- Manually
- As a Daemon
- Integrated with XAG agent



Note:

If the Service Manager is registered as a system daemon, then the Service Manager, Administration Server (AS), Distribution Server (DS), Receiver Server (RS), and the Performance Metrics Server are automatically started when the host is (re)started.

Oracle recommends the usage of a secure configuration within Oracle GoldenGate MA. There are two options for setting up a secure MA deployment:

- Run MA on loopback address and front it with an HTTPS reverse proxy (nginx). See Reverse Proxy Support.
 - Interoperability between Oracle GoldenGate Classic and Oracle GoldenGate MA is configured through the `ogg` protocol using data pump Extract from Oracle GoldenGate Classic with `SOCKSPROXY`.
- Run Oracle GoldenGate MA with TLS version 1.2 enabled on all services.

For more information on setting up the Service Manager as a daemon service, see [Select Service Manager Options](#)

Quick Tour of the Service Manager

When you complete the Oracle GoldenGate MA installation, the Service Manager opens up at the specified URL. This page acts as an access point for performing deployment, configuring the Administration Server, Distribution Server, Receiver Server, Performance Metrics Server, and the Admin Client.

The Service Manager home page is a dashboard where you can see the services that have been deployed and access inventory and configuration information pertaining to your deployments. You can also view the status of your deployments, and start and stop services.

Now, that you have an overview of the Service Manager, let's go through some of the actions you can perform using the Service Manager home page.

Action	Task
View the service status	Review Status Changes

Action	Task
Start and stop deployments	Starting and Stopping Deployments and Services
Access various servers	<p>You can click the respective links to access the following:</p> <ul style="list-style-type: none"> Administration Server to add, modify, and delete Extracts and Replicats. Distribution Server to add, modify, and delete Paths Performance Metrics Server to Review Messages and Review Status Changes Receiver Server to view details of the path, including path network statistics and file I/O statistics.
Access details for Administration Server, Distribution Server, Performance Metrics Server, and Receiver Server	Click Details for the server for which you need to see the details. See View and Edit Services Configuration
Application Navigation pane	Click the icon to expand and access the Service Manager or the Diagnosis home pages.

How to Start and Stop the Service Manager

The start and stop process of the Service Manager within Oracle GoldenGate Microservices Architecture is different based on how the Service Manager is configured within your environment.

The following provide context on how the start and stop processes can be done for the Service Manager:

- If the Service Manager is configured in manual mode then there are scripts in the `$DEPLOYMENT_BASE/ServiceManager/bin` directory that can be run to start or stop the Service Manager.

Run the scripts to start or stop the Service Manager from the following locations:

- To start the Service Manager: `OGG_Deployment_Home/bin/startSM.sh`
- To stop the Service Manager: `OGG_Deployment_Home/bin/stopSM.sh`

- If the Service Manager is configured as a daemon, the scripts required to start or stop for manual interaction are not created. The operating system is responsible for starting or stopping the Service Manager.

For OEL 6:

```
stop/start/status for Service Manager
/etc/init.d/OracleGoldenGate start
/etc/init.d/OracleGoldenGate stop
/etc/init.d/OracleGoldenGate status
```

For OEL 7:

```
systemctl start OracleGoldenGate
systemctl status OracleGoldenGate
systemctl stop OracleGoldenGate
```

- If the Service Manager is configured to run with the XAG agent in an Oracle Cluster Ready Service (CRS); then the start and stop process is handled by the CRS stack.

How to Change Deployment Details and Configuration

You can review and change the selected service (server) configuration.

Details Tab

Use to review the selected deployment configuration. All the deployment directories that you configured with the Configuration Assistant are displayed. For Oracle database, the only directory that you can edit is the Oracle GoldenGate home (`OGG_HOME`). This allows you to use a different installation than the one you originally configured.

Configuration Tab

Use to review and change the selected deployment environment variables. The environment variables that you configured for your deployment are displayed. You can add new variables, modify existing variables, and delete selected variables.

When using Oracle GoldenGate Microservices on an AIX operating with Oracle database RU11 and higher, the `AIXTHREAD_STK` value needs to be set to atleast 1048576 (1 MB). You can set the `AIXTHREAD_STK` value from this tab, as follows:

Add an environment variable for `AIXTHREAD_STK` for the deployment.

Restart the deployment.

Check the Extract report file to these updates.

The Extract thread `IXAsyncTrans` is set to a minimum size of 2M.

The default stack size on AIX is 196,608 bytes for 64-bit applications.

Certificates

Use this tab to manage certificates for client and CA certificates. See *Securing Deployments* in the Oracle GoldenGate Security Guide for details.

How to Interpret the Log Information

You can review all of the messages logged for your Service Manager with this page.

Using the Table

An updated log of Extract and Replicat server messages is displayed. You can sort the list by date or severity by clicking on the adjacent arrow. Also, you can refresh this log and choose how many pages you want to view.

To search, you select Date, Severity, or Message, and then select the appropriate options to construct your search.

Notice the **Notifications** tab at the bottom of the page. It displays server messages, which are not updated in the log due to transaction errors. For example, failure to log in to the database using the database credentials.

How to Enable and Use Debug Logging

You can enable debug logging and download debug log files from this page.

Enabling Debug Logging:

To enable debug logging:

1. Click the Debug Log option from the Navigation Pane of the Service Manager page.
2. Click the Enable Debug Log option to start logging debug information.

Using the Debug Log

You can use the access and use the debug log file from this page:

1. Click the **Download Log File** option to save a local copy of the debug log
2. Click the **Load Debug Log File** option to view the debug log on this page.
3. Search for specific entries in the debug log using the **Search By** box, if required. You can click **Refresh** to get the latest log information, if it doesn't get refreshed automatically.

How to Start and Stop Service Manager and Deployments

The Service Manager is the central hub from where you can start and stop deployments and other microservices such as Administration Server, Distribution Server, Performance Metrics Server, and Receiver Server.

Using Service Manager Start or Stop a Deployment



Note:

If Oracle GoldenGate Service Manager is registered as a system daemon, then the Service Manager along with the other servers, are automatically started when the host is (re)started.

Using Scripts to Start and Stop a Deployment

The Service Manager deployment include startup and shutdown scripts (`startSM.sh` and `stopSM.sh`) for starting and stopping the deployment locally from the command line.

Here are the steps to access and run the scripts:

1. Ensure that your environment variables, mainly the `ETC_HOME` and the `VAR_HOME`, are set up correctly. See [Add a Deployment](#) for environment variable setup.
2. Navigate to `DEPLOYMENT_HOME/bin` directory for the Service Manager.



Note:

If you selected to run the Service Manager as a system daemon, then these script files will not be in this location. Instead, the bin directory would contain the file, `oggInst.loc`, which is used to register the Service Manager as a daemon.

3. Run the following command to stop the Service Manager:

```
./stopSM.sh
```

4. Run the following command, to start or restart the Service Manager:

```
./startSM.sh
```

Remove a Deployment

Learn about removing a deployment.

Before Removing the Deployment

Removing a deployment is not the same as removing a Service Manager. When you remove a deployment, it doesn't imply that the Service Manager would also need to be removed as there could be multiple deployments added to the same Service Manager.

You can remove a deployment using the Oracle GoldenGate Configuration Assistant (OGGCA) wizard.



Note:

When you remove a deployment or uninstall Oracle GoldenGate MA, the system does not automatically stop processes. As a result, you may have to stop processes associated with the deployment and you must clean files manually.

Before removing a deployment, stop the deployment, its associated microservices, and ER processes.

Start OGGCA to Remove Deployment

To start the deployment removal process, follow these steps:

1. Run the OGGCA wizard from the following location:

```
cd $OGG_HOME/bin  
./OGGCA.sh
```

2. Select **Existing Service Manager** from the **Select Service Manager Options** screen. Click **Next**.
3. Select **Remove Existing Oracle GoldenGate Deployment** from the **Configuration Options** screen. Click **Next**.
4. Select the deployment you need to remove from the **Deployment Name** list box.
5. Select the **Delete Deployment Files from Disk** check box if you want to remove all the deployment files (including configuration files) from the host server. These configuration files are usually located in the `/etc` and `/conf` directories.
6. Enter the Administration account user name and password for the Service Manager administrator.

7. Enter the Administration account user name and password for the Deployment administrator click **Next**.
8. On the Summary page, see the list of settings that would be deleted with the deployment and click **Finish**.

Remove the Service Manager

Learn about removing the Service Manager.

Start OGGCA to Remove the Service Manager

The option to remove the Service Manager is available in OGGCA, only if there are no available deployments to remove. To remove the Service Manager:

1. Run the OGGCA wizard from the /bin directory of Oracle GoldenGate home:

```
cd $OGG_HOME/bin
./oggca.sh
```
2. Select **Existing Service Manager** from the **Select Service Manager Options** screen. Click **Next**.
3. Select the **Service Manager** from the drop down list.
4. Select **Remove Service Manager Deployment** from the **Configuration Options** screen.
5. Click **Finish** to remove the Service Manager.

Files to be Removed Manually After Removing Deployment

It's mandatory to delete some files manually only in case there's a Service Manager registered but you have to unregister it and register a new one. To remove files manually, you must have `root` or `sudo` privileges. The files to be deleted include:

Operating System	Files to be Removed Manually to Unregister an Existing Service Manager
Linux 6	<ul style="list-style-type: none">• /etc/init.d/OracleGoldenGate• /etc/rc.d/*OracleGoldenGate• /etc/rc*.d/*OracleGoldenGate• /etc/oggInst.loc
Linux 7	/etc/systemd/system/ OracleGoldenGate.service

The following commands are executed to stop the Service Manager:

```
systemctl stop OracleGoldenGate
systemctl disable OracleGoldenGate *
```

**Note:**

If the Service Manager is not registered as a service (with or without the integration with XAG), OGGCA stops the Service Manager deployment, otherwise, a script called `unregisterServiceManager` is created. When executed by the user, it runs the `systemctl` commands and deletes the mentioned files.

View and Edit Services Configuration

The services configuration and restart options for Administration Server, Distribution Server, Performance Metrics Server, and Receiver Server can be viewed and edited from the Services Manager.

You can access the services configuration for each of the servers, from the Service Manager home page. Click the Details button for the server that you need to check the service configuration for. The Service Configuration page is displayed. This page allows you to view and edit the service configuration and the restart options for the corresponding server. The configuration and restart options for all the servers are the same.

The following table explains the Service Configuration and Restart Options on the Services Configuration page.

Service Configuration Options	Description
Port	Port Number for the corresponding server
Enable Legacy Protocol	Enables legacy communication for services that are compatible.
Enabled Async Operation	Enables asynchronous RESTful API method execution
Default Sync Wait	The default time a service will wait before responding with an asynchronous REST API response
Enabled Task Manager	Enable task management for services that provide it.
U-Mask	File mode creation mask
Quiet	Starts the service in quiet mode.
Enabled	Indicates that the service is managed by Service Manager.
Status	Indicates that the service is running.
Restart Options	Description
Enabled	If set to true, then it restart a task if it gets terminated.
On Success	If set to false, then the task is only restarted if it fails.
Delay	The time (in minutes) to pause between discovering that a process is terminated abruptly and restarting it.
Retries	The maximum number of trials to restart the service, before aborting the retry effort.

Window	The time interval in which the retries are counted. The default is 120 minutes.
Disable on Failure	If set to true, the task is disabled after it fails all execution attempts in an execution window.

4

Prepare

Learn about the tasks for preparing databases for Oracle GoldenGate and prerequisites for connecting Oracle GoldenGate to databases before beginning the configuration of Extract and Replicat processes.

Prepare Oracle Database

Prepare Oracle database for Oracle GoldenGate by enabling Oracle GoldenGate on the database side, enabling supplemental logging, configuring database connections for multitenant container databases, managing server resources, and various other tasks, as required.

Prepare Database Users and Privileges for Oracle

Learn about creating database users and assigning privileges for Oracle GoldenGate for Oracle.

Grant User Privileges for Oracle Database 21c and Lower

The user privileges that are required for connecting to Oracle database from Oracle GoldenGate depend on the type of user.

Privileges should be granted depending on the actions that the user needs to perform as the GoldenGate Administrator User on the source and target databases. For example, to grant DML operation privileges to insert, update, and delete transactions to a user, use the `GRANT ANY INSERT/UPDATE/DELETE` privileges and to further allow users to work with tables and indexes as part of DML operations, use the `GRANT CREATE/DROP/ALTER ANY TABLE/INDEX` privileges.

If the GoldenGate Administrator user has the DBA role, additional object privileges are not needed. However, there might be security constraints granting the DBA role to the GoldenGate Administration user. The DBA role is not necessarily required for Oracle GoldenGate.

If there are many objects being replicated, you might consider using the ANY privilege for DML and DDL operations. This simplifies the provision of privileges to the GoldenGate Administrator users, as you only need to grant a few privileges depending on the database operations.

The following table describes some of the essential privileges for GoldenGate Administrator user for Oracle database. For explanation purposes, the table uses `c##ggadmin` as an example of a common user for a multitenant container database and `ggadmin` as the pluggable database (PDB) user. `PDBEAST` and `PDBWEST` are used as examples of PDB names.

The following table describes the essential privileges for GoldenGate Administrator user for using Oracle GoldenGate with on source and target Oracle databases:

Privilege	Extract	Replicat All Modes	Purpose
RESOURCE	Yes	Yes	<p>Required to create objects</p> <p>In Oracle Database 12cR1 and later, instead of RESOURCE, grant the following privilege:</p> <pre>ALTER USER user QUOTA {size UNLIMITED} ON tablespace;</pre>
CONNECT	Yes	Yes	<p>Common user SYSTEM connects to the root container. This privilege is essential when the DBA role is not assigned to the user.</p> <p>See an example of Permissions granted to an Oracle multitenant database common user.</p>
CREATE PROCEDURE	Yes	Yes	Required to add heartbeat tables.
CREATE SESSION	Yes	Yes	Required to connect to the database.
CREATE VIEW	Yes	Yes	<p>Required to add the heartbeat table view.</p> <p>If you want to be specific to each object, you can also provide the privileges for each object individually. You may consider creating a specific database role to maintain such privileges.</p>
ALTER SYSTEM	Yes	Yes	Perform administrative changes, such as enabling logging.
ALTER USER	Yes	Yes	Required for multitenant architecture and GGADMIN should be a valid Oracle GoldenGate administrator schema.
EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE ('REPUSER', CONTAINER=>'PDBEAST');	Yes	Yes	<ul style="list-style-type: none"> Required for Autonomous Databases (ATP and ADW) Extract and Replicat. Extracts in the root container (CDB\$ROOT)) might require a value of ALL or a specific PDB (example: pdbeast). Grant privileges for Extract and Replicat users. See Example: Grant privileges using the DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE package Grant privileges to capture from Virtual Private Database Grants privileges to capture redacted data

Privilege	Extract	Replicat All Modes	Purpose
Grant DV_GOLDENGATE_ADMIN and DV_GOLDENGATE_REDO_ACCESS privileges connected as SYS user to the Extract and the Replicat user.	Yes	Yes	Capture from Data Vault. See Privileges for Capturing from Oracle Data Vault .
Grant Replicat privileges in DBMS_MACADM.ADD_AUTH_TO_REALM if applying to a realm.	NA	Yes	Capture from Data Vault. See Privileges for Capturing from Oracle Data Vault .
INSERT, UPDATE, DELETE on target tables	NA	Yes	Apply replicated DML to target objects. See Details of Support for Objects and Operations in Oracle DML .
GRANT INSERT ANY TO...	NA	Yes	Grant these privileges to the Replicat user, instead of granting INSERT, UPDATE, DELETE to every table, if replicating every table.
GRANT UPDATE ANY TO...			
GRANT DELETE ANY TO...			
If DDL replication is performed, grant the following as Database Vault owner: EXECUTE DBMS_MACADM.AUTHORIZE_DDL('GGADMIN USER', 'SCHEMA FOR DDL');	No	No	Capture from Data Vault. See Privileges for Capturing from Oracle Data Vault .
DDL privileges on target objects (if using DDL support)	NA	Yes	Issue replicated DDL on target objects. See Details of Support for Objects and Operations in Oracle DDL .
GRANT [CREATE ALTER DROP] ANY [TABLE INDEX VIEW PROCEDURE] to GGADMIN;	Yes	Yes	Grants privileges for DDL Replication for tables.
CREATE ANY TABLE	Yes	Yes	Grants privileges for creating table in any schema. To allow creating tables only in a specific schema, use the CREATE TABLE privilege.
CREATE ANY VIEW	Yes	Yes	Grants privileges to create view in any database schema. To allow creating views in a specific schema, use the CREATE VIEW privilege.
SELECT ANY DICTIONARY	Yes	Yes	Allow all privileges to work properly on dictionary tables.

Example: Permissions granted for the Oracle database common user

Privileges granted for the Oracle database common user, which is c##ggadmin in the following example:

```
CREATE USER c##ggadmin IDENTIFIED BY passw0rd CONTAINER=all DEFAULT
TABLESPACE GG_DATA TEMPORARY TABLESPACE temp;
GRANT RESOURCE to c##ggadmin;
GRANT CREATE SESSION to c##ggadmin;
GRANT CREATE VIEW to c##ggadmin;
GRANT CREATE TABLE to c##ggadmin;
GRANT CONNECT to c##ggadmin CONTAINER=all;
GRANT DV_GOLDENGATE_ADMIN; --- for data vault user
GRANT DV_GOLDENGATE_REDO_ACCESS; --- for data vault user
GRANT ALTER SYSTEM to c##ggadmin;
GRANT ALTER USER to c##ggadmin;
ALTER USER c##ggadmin SET CONTAINER_DATA=all CONTAINER=current;
ALTER USER c##ggadmin QUOTA unlimited ON GG_DATA;
GRANT SELECT ANY DICTIONARY to c##ggadmin;
GRANT SELECT ANY TRANSACTION to c##ggadmin;
EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE('c##ggadmin');
```

In this example, DBA privilege is not provided. If privileges are missing, then the DBA has to grant necessary privileges additionally.

Privileges granted for PDB user ggadmin are provided in the following example:

```
ALTER SESSION SET CONTAINER=dbwest;
CREATE USER ggadmin IDENTIFIED BY PASSWORD CONTAINER=CURRENT;
GRANT CONNECT, RESOURCE, DBA TO ggadmin CONTAINER=CURRENT;
GRANT CREATE SESSION TO ggadmin CONTAINER=CURRENT;
EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE('ggadmin');
```

**Note:**

Granting DBA role is not mandatory for every user. Privileges should be granted depending on the actions that the user needs to perform on the database. For example, to grant DML operation privileges to insert, update, and delete transactions to ggadmin, use the GRANT ANY INSERT/UPDATE/DELETE privileges and to further allow users to work with tables and indexes as part of DML operations, use the GRANT CREATE/DROP/ALTER ANY TABLE/INDEX privileges.

Example: Grant privileges using the DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE package

This procedure grants the privileges needed by a user to be an Oracle GoldenGate administrator. The following example grants explicit privileges for Extract on Oracle multitenant database:

```
BEGIN
DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE
(GRANTEE => 'c##ggadmin', PRIVILEGE_TYPE => 'CAPTURE',
```

```
GRANT_SELECT_PRIVILEGES => TRUE, DO_GRANTS => TRUE, CONTAINER => 'ALL' );
END;
```

See `DBMS_GOLDENGATE_AUTH` in *Oracle Database PL/SQL Packages and Types Reference* for more information.

About the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE` Package

Most of the privileges that are needed for Extract and Replicat to operate are granted through the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE` package.

The first example is the default, which grants to both Extract and Replicat. The second shows how to explicitly grant to either Extract or Replicat (in this case, Extract).

```
GRANT_ADMIN_PRIVILEGE ('ggadmin')
GRANT_ADMIN_PRIVILEGE ('ggadmin','exte');
```

The following example shows Extract on Oracle 12c multitenant database:

```
BEGIN
DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE
(GRANTEE => 'c##ggadmin', PRIVILEGE_TYPE => 'CAPTURE',
 GRANT_SELECT_PRIVILEGES => TRUE, DO_GRANTS => TRUE, CONTAINER => 'ALL' );
END;
```

Optional Grants for `dbms_goldengate_auth.grant_admin_privilege`

This procedure grants the privileges needed by a user to be a Oracle GoldenGate administrator. See `DBMS_GOLDENGATE_AUTH` in *Oracle Database PL/SQL Packages and Types Reference* for more information.

Optional Grants for `dbms_goldengate_auth.grant_admin_privilege`

Privileges for Capturing from Oracle Data Vault

Grant the following privileges connected as `SYS` user in Oracle database. These privileges are set for Extract and Replicat user credentials:

- `EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE ('userID','*',`
`GRANT_OPTIONAL_PRIVILEGES=> '*');`
`GRANT DV_GOLDENGATE_ADMIN, DV_GOLDENGATE_REDO_ACCESS to userID;`
- Grant Replicat the privileges in `DBMS_MACADM.ADD_AUTH_TO_REALM` if applying to a realm.
Connect as Database Vault owner and execute the following scripts:

```
BEGIN
DVSYS.DBMS_MACADM.ADD_AUTH_TO_REALM(
REALM_NAME => 'Oracle Default Component Protection Realm',GRANTEE =>
'userID',AUTH_OPTIONS => 1) ;
END ;
```

```
/
EXECUTE_DBMS_MACADM.AUTHORIZE_DDL('SYS', 'SYSTEM');
```

- For DDL replication, grant the following as the Database Vault owner:

```
EXECUTE_DBMS_MACADM.AUTHORIZE_DDL
('userID', 'SCHEMA FOR DDL');
```

Configuring Connections for Integrated Processes

If you will be using integrated capture and integrated Replicat, each requires a dedicated server connection in the `tnsnames.ora` file.

You direct the processes to use these connections with the `USERID` or `USERIDALIAS` parameter in the Extract and Replicat parameter files when you configure those processes.

The following is an example of the dedicated connection required for integrated capture (Extract) and integrated Replicat.

```
TEST =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP) (HOST = test2) (PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = test)
    )
  )
```

The following are the security options for specifying the connection string in the Extract or Replicat parameter file.

Password encryption method:

```
USERID intext@test, PASSWORD mypassword
```

Credential store method:

```
USERIDALIAS ext
```

In the case of `USERIDALIAS`, the alias `ext` is stored in the Oracle GoldenGate credential store with the actual connection string, as in the following example:

```
AdminClient INFO CREDENTIALSTORE DOMAIN support
Domain: Support
Alias: ext
Userid: intext@test
```

Configuring Logging Properties

Oracle GoldenGate relies on the redo logs to capture the data that it needs to replicate source transactions. The Oracle redo logs on the source system must be configured properly before you start Oracle GoldenGate processing.

This section addresses the following logging levels that apply to Oracle GoldenGate. Which logging level that you use is dependent on the Oracle GoldenGate feature or features that you are using.

**Note:**

Redo volume is increased as the result of this required logging. You can wait until you are ready to start Oracle GoldenGate processing to enable the logging.

This table shows the Oracle GoldenGate use cases for the different logging properties.

Logging option	GGSCI command	What it does	Use case
Forced logging mode	ALTER DATABASE FORCE LOGGING;	Forces the logging of all transactions and loads.	Strongly recommended for all Oracle GoldenGate use cases. <code>FORCE LOGGING</code> overrides any table-level <code>NOLOGGING</code> settings.
Minimum database-level supplemental logging	ALTER DATABASE ADD SUPPLEMENTAL LOG DATA	Enables minimal supplemental logging to add row-chaining information to the redo log.	Required for all Oracle GoldenGate use cases
Schema-level supplemental logging, default setting See Enabling Schema-level Supplemental Logging .	ADD SCHEMATRANDATA	Enables unconditional supplemental logging of the primary key and conditional supplemental logging of unique key(s) and foreign key(s) of all tables in a schema. All of these keys together are known as the <i>scheduling columns</i> .	Enables the logging for all current and future tables in the schema. If the primary key, unique key, and foreign key columns are not identical at both source and target, use <code>ALLCOLS</code> .
Schema-level supplemental logging with unconditional logging for all supported columns. (See Enabling Schema-level Supplemental Logging for non-supported column types.)	ADD SCHEMATRANDATA with <code>ALLCOLS</code> option	Enables unconditional supplemental logging of all of the columns in a table, for all of the tables in a schema.	Used for bidirectional and active-active configurations where all column values are checked, not just the changed columns, when attempting to perform an update or delete. This takes more resources though allows for the highest level of real-time data validation and thus conflict detection. This method should also be used if they are going to be using the <code>HANDLECOLLISIONS</code> parameter for initial loads.
Schema-level supplemental logging, minimal setting	ADD SCHEMATRANDATA with <code>NOSCHEDULINGCOLS</code> option	Enables unconditional supplemental logging of the primary key and all valid unique indexes of all tables in a schema.	Use only for nonintegrated Replicat. This is the minimum required schema-level logging.

Logging option	GGSCI command	What it does	Use case
Table-level supplemental logging with built-in support for integrated Replicat See Enabling Table-level Supplemental Logging	ADD TRANDATA	Enables unconditional supplemental logging of the primary key and conditional supplemental logging of unique key(s) and foreign key(s) of a table. All of these keys together are known as the <i>scheduling columns</i> .	Required for all Oracle GoldenGate use cases unless schema-level supplemental logging is used. If the primary key, unique key, and foreign key columns are not identical at both source and target, use ALLCOLS.
Table-level supplemental logging with unconditional logging for all supported columns. (See Enabling Table-level Supplemental Logging for non-supported column types.)	ADD TRANDATA with ALLCOLS option	Enables unconditional supplemental logging of all of the columns of the table.	Used for bidirectional and active-active configurations where all column values are checked, not just the changed columns, when attempting to perform an update or delete. This takes more resources though allows for the highest level of real-time data validation and thus conflict detection. It can also be used when the source and target primary, unique, and foreign keys are not the same or are constantly changing between source and target.
Table-level supplemental logging, minimal setting	ADD TRANDATA with NOSCHEDULINGCOLS option	Enables unconditional supplemental logging of the primary key and all valid unique indexes of a table.	Use for nonintegrated Replicat and non-parallel Replicat. This is the minimum required table-level logging.

**Note:**

Oracle Databases must be in ARCHIVELOG mode so that Extract can process the log files.

Enabling Minimum Database-level Supplemental Logging

Oracle strongly recommends putting the Oracle source database into forced logging mode. Forced logging mode forces the logging of all transactions and loads, overriding any user or storage settings to the contrary. This ensures that no source data in the Extract configuration gets missed.

In addition, minimal supplemental logging, a database-level option, is required for an Oracle source database when using Oracle GoldenGate. This adds row chaining information, if any exists, to the redo log for update operations.

**Note:**

Database-level primary key (PK) and unique index (UI) logging is only discouraged if you are replicating a subset of tables. You can use it with Live Standby, or if Oracle GoldenGate is going to replicate all tables, like to reduce the downtime for a migration or upgrade.

Perform the following steps to verify and enable, if necessary, minimal supplemental logging and forced logging.

1. Log in to SQL*Plus as a user with `ALTER SYSTEM` privilege.
2. Issue the following command to determine whether the database is in supplemental logging mode and in forced logging mode. If the result is `YES` for both queries, the database meets the Oracle GoldenGate requirement.

```
SELECT supplemental_log_data_min, force_logging FROM v$database;
```

3. If the result is `NO` for either or both properties, continue with these steps to enable them as needed:

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;  
SQL> ALTER DATABASE FORCE LOGGING;
```

4. Issue the following command to verify that these properties are now enabled.

```
SELECT supplemental_log_data_min, force_logging FROM v$database;
```

The output of the query must be `YES` for both properties.

5. Switch the log files.

```
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

Enabling Schema-level Supplemental Logging

Oracle GoldenGate supports schema-level supplemental logging. Schema-level logging is required for an Oracle source database when using the Oracle GoldenGate DDL replication feature. In all other use cases, it is optional, but then you must use table-level logging instead (see [Enabling Table-level Supplemental Logging](#)).

By default, schema-level logging automatically enables unconditional supplemental logging of the primary key and conditional supplemental logging of unique key(s) and foreign key(s) of all tables in a schema. Options enable you to alter the logging as needed.

**Note:**

Oracle strongly recommends using schema-level logging rather than table-level logging, because it ensures that any new tables added to a schema are captured if they satisfy wildcard specifications. This method is also recommended because any changes to key columns are automatically reflected in the supplemental log data too. For example, if a key changes, there is no need to issue `ADD TRANDATA`.

Perform the following steps on the source system to enable schema-level supplemental logging.

1. Start the command line on the source system.
2. Issue the `DBLOGIN` command with the alias of a user in the credential store who has privilege to enable schema-level supplemental logging.

```
DBLOGIN USERIDALIAS alias
```

See `USERIDALIAS` in *Parameters and Functions Reference for Oracle GoldenGate* for more information about `USERIDALIAS` and additional options.

3. When using `ADD SCHEMATRANDATA` or `ADD TRANDATA` on a multitenant database, you can either log directly into the PDB and perform the command. Alternately, if you are logging in at the root level (using a `C##` user), then you must include the PDB. Issue the `ADD SCHEMATRANDATA` command for each schema for which you want to capture data changes with Oracle GoldenGate.

```
ADD SCHEMATRANDATA pdb.schema [ALLCOLS | NOSCHEDULINGCOLS]
```

Where:

- Without options, `ADD SCHEMATRANDATA` schema enables the unconditional supplemental logging on the source system 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 default is optional to support nonintegrated Replicat but is required to support integrated Replicat because 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 [Types of Replicat](#).
- `ALLCOLS` can be used to enable the unconditional supplemental logging of all of the columns of a table and applies to all current and future tables in the given schema. Use to support integrated Replicat when the source and target tables have different scheduling columns. (*Scheduling columns* are the primary key, the unique key, and the foreign key.)
- `NOSCHEDULINGCOLS` logs only the values of the primary key and all valid unique indexes for existing tables in the schema and new tables added later. This is the minimal required level of schema-level logging and is valid only for Replicat in nonintegrated mode.

In the following example, the command enables default supplemental logging for the `hr` schema.

```
ADD SCHEMATRANDATA pdbeast.hr ALLCOLS
```

In the following example, the command enables the supplemental logging only for the primary key and valid unique indexes for the `HR` schema.

```
ADD SCHEMATRANDATA pdbeast.hr NOSCHEDULINGCOLS
```

Enabling Table-level Supplemental Logging

Enable table-level supplemental logging on the source system in the following cases:

- To enable the required level of logging when not using schema-level logging (see [Enabling Table-level Supplemental Logging](#)[Enabling Schema-level Supplemental Logging](#)). Either schema-level or table-level logging must be used. By default, table-level logging automatically enables unconditional supplemental logging of the primary key and conditional supplemental logging of unique key(s) and foreign key(s) of a table. Options enable you to alter the logging as needed.
- To prevent the logging of the primary key for any given table.
- To log non-key column values at the table level to support specific Oracle GoldenGate features, such as filtering and conflict detection and resolution logic.
- If the key columns change on a table that only has table-level supplemental logging, you must perform `ADD TRANDATA` on the table prior to allowing any DML activity on the table.

Perform the following steps on the source system to enable table-level supplemental logging or use the optional features of the command.

1. Run the command line on the source system.
2. Issue the `DBLOGIN` command using the alias of a user in the credential store who has privilege to enable table-level supplemental logging.

```
DBLOGIN USERIDALIAS alias
```

See `USERIDALIAS` in *Parameters and Functions Reference for Oracle GoldenGate* for more information about `DBLOGIN` and additional options.

3. Issue the `ADD TRANDATA` command.

```
ADD TRANDATA [PDB.]schema.table [, COLS (columns)] [, NOKEY] [, ALLCOLS |  
NOSCHEDULINGCOLS]
```

Where:

- *PDB* is the name of the root container or pluggable database if the table is in a multitenant container database.
- *schema* is the source schema that contains the table.
- *table* is the name of the table. See *Specifying Object Names in Oracle GoldenGate Input* in *Administering Oracle GoldenGate* for instructions for specifying object names.
- `ADD TRANDATA` without other options automatically enables unconditional supplemental logging of the primary key and conditional supplemental logging of unique key(s) and foreign key(s) of the 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 default is optional to support nonintegrated Replicat (see also `NOSCHEDULINGCOLS`) but is required to support integrated Replicat because 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 [Types of Replicat](#).
- `ALLCOLS` enables the unconditional supplemental logging of all of the columns of the table. Use to support integrated Replicat when the source and target tables have different scheduling columns. (*Scheduling columns* are the primary key, the unique key, and the foreign key.)

- `NOSCHEDULINGCOLS` is valid for Replicat in nonintegrated mode only. It issues an `ALTER TABLE` command with an `ADD SUPPLEMENTAL LOG DATA ALWAYS` clause that is appropriate for the type of unique constraint that is defined for the table, or all columns in the absence of a unique constraint. This command satisfies the basic table-level logging requirements of Oracle GoldenGate when schema-level logging will not be used. See [Ensuring Row Uniqueness in Source and Target Tables](#) for how Oracle GoldenGate selects a key or index.
 - `COLS columns` logs non-key columns that are required for a `KEYCOLS` clause or for filtering and manipulation. The parentheses are required. These columns will be logged in addition to the primary key unless the `NOKEY` option is also present.
 - `NOKEY` prevents the logging of the primary key or unique key. Requires a `KEYCOLS` clause in the `TABLE` and `MAP` parameters and a `COLS` clause in the `ADD TRANDATA` command to log the alternate `KEYCOLS` columns.
4. If using `ADD TRANDATA` with the `COLS` option, create a unique index for those columns on the target to optimize row retrieval. If you are logging those columns as a substitute key for a `KEYCOLS` clause, make a note to add the `KEYCOLS` clause to the `TABLE` and `MAP` statements when you configure the Oracle GoldenGate processes.

Enabling Oracle GoldenGate in the Database

The database services required to support Oracle GoldenGate capture and apply must be enabled explicitly for all Oracle database versions. This is required for Extract and all Replicat modes.

To enable Oracle GoldenGate, set the following database initialization parameter. All instances in Oracle RAC must have the same setting.

```
ENABLE_GOLDENGATE_REPLICATION=true
```

This parameter alters the `DBA_FEATURE_USAGE_STATISTICS` view. For more information about this parameter, see Initialization Parameters.

Configuring Oracle GoldenGate in a Multitenant Container Database

This chapter contains additional configuration instructions when configuring Oracle GoldenGate in a multitenant container database (CDB).

Using the Root Container Extract from PDB

To capture from a multitenant database, you must use an Extract that is configured at the root level using a `c##` account. To apply data into a multitenant database, a separate Replicat is needed for each PDB, because a Replicat connects at the PDB level and doesn't have access to objects outside of that PDB.

One Extract group can capture from multiple pluggable databases to a single trail. In the parameter file, source objects must be specified in `TABLE` and `SEQUENCE` statements with their fully qualified three-part names in the format of `container.schema.object`.

As an alternative to specifying three-part names, you can specify a default pluggable database with the `SOURCECATALOG` parameter, and then specify only the `schema.object` in subsequent `TABLE` or `SEQUENCE` parameters. You can use multiple instances of this configuration to handle multiple source pluggable databases. For example:

```

SOURCECATALOG pdb1
TABLE phoenix.tab;
SEQUENCE phoenix.seq;
SOURCECATALOG pdb2
TABLE dallas.tab;
SEQUENCE dallas.seq;

```

Applying to Pluggable Databases

Replicat can only connect and apply to one pluggable database. To specify the correct one, use a SQL*Net connect string for the database user that you specify with the `USERID` or `USERIDALIAS` parameter. For example: `GGADMIN@FINANCE`. In the parameter file, specify only the `schema.object` in the `TARGET` portion of the `MAP` statements. In the `MAP` portion, identify source objects captured from more than one pluggable database with their three-part names or use the `SOURCECATALOG` parameter with two-part names. The following is an example of this configuration.

```

SOURCECATALOG pdb1
MAP schema_1.tab, TARGET 1;
MAP schema_1.seq, TARGET 1;
SOURCECATALOG pdb2
MAP schema_2.tab, TARGET 2;
MAP schema_2.seq, TARGET 2;

```

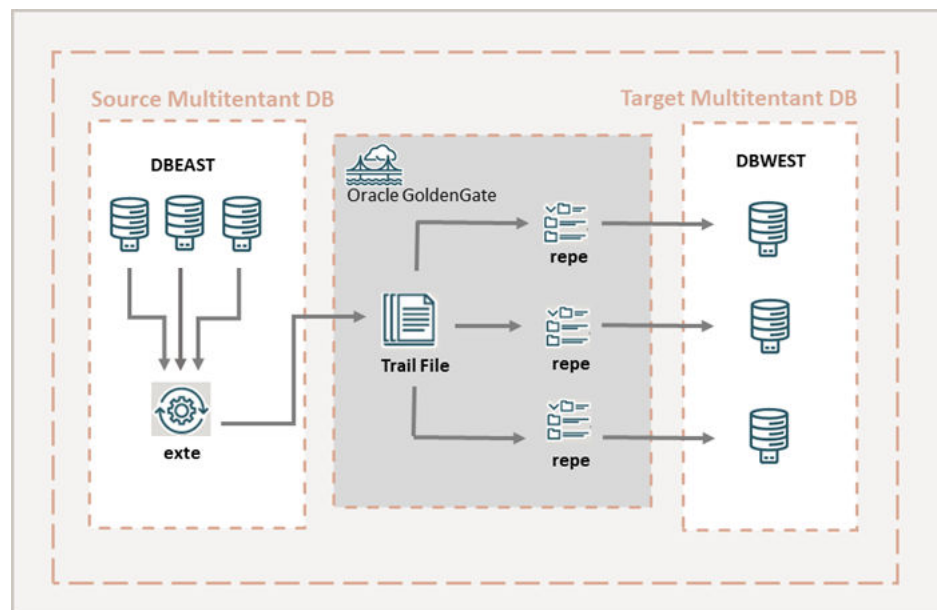
The following is an example *without* the use of `SOURCECATALOG` to identify the source pluggable database. In this case, the source objects are specified with their three-part names.

```

MAP pdb1.schema_1.tab, TARGET 1;
MAP pdb1.schema_1.seq, TARGET 1;

```

To configure replication from multiple source pluggable databases to multiple target pluggable databases, you can configure parallel Extract and Replicat streams, each handling data for one pluggable database. Alternatively, you can configure one Extract capturing from multiple source pluggable databases, which writes to one trail that is read by multiple Replicat groups, each applying to a different target pluggable database. Yet another alternative is to use one Extract writing to multiple trails, each trail read by a Replicat assigned to a specific target pluggable database :



Excluding Objects from the Configuration

To exclude pluggable databases, schemas, and objects from the configuration, you can use the `CATALOGEXCLUDE`, `SCHEMAEXCLUDE`, `TABLEEXCLUDE`, `MAPEXCLUDE`, and `EXCLUDEWILDCARDOBJECTSONLY` parameters.

Requirements for Configuring Container Databases for Oracle GoldenGate

This topic describes the special requirements that apply to replication to and from multitenant container databases.

The requirements are:

- The different pluggable databases in the multitenant container database can have different character sets. Oracle GoldenGate captures data from any multitenant database with different character sets into one trail file and replicates the data without corruption due to using different character sets.
- Extract must operate in integrated capture mode. See [Deciding Which Capture Method to Use](#) for more information about Extract capture modes. Replicat can operate in any of its modes.
- Extract must connect to the root container (`cdb$root`) as a common user in order to interact with the logmining server. To specify the root container, use the appropriate SQL*Net connect string for the database user that you specify with the `USERID` or `USERIDALIAS` parameter. For example: `C##GGADMIN@FINANCE`. See [Establishing Oracle GoldenGate Credentials](#) for how to create a user for the Oracle GoldenGate processes and grant the correct privileges.
- To support source CDB 12.2, Extract must specify the trail format as release 12.3. Due to changes in the redo logs, to capture from a multitenant database that is Oracle 12.2 or higher, the trail format release must be 12.3 or higher.
- The `dbms_goldengate_auth.grant_admin_privilege` package grants the appropriate privileges for capture and apply within a multitenant container database. This includes the `container` parameter, which must be set to `ALL`, as shown in the following example:

```
exec dbms_goldengate_auth.grant_admin_privilege('C##GGADMIN',container=>'all')
```
- DDL replication works as a normal replication for multitenant databases. However, DDL on the root container should not be replicated because Replicats must not connect to the root container, only to PDBs.

FLUSH SEQUENCE for Multitenant Database

`FLUSH SEQUENCE` must be issued at the PDB level, so the user will need to create an Oracle GoldenGate user in each PDB that they wish to do sequence replication for, and then use `DBLOGIN` to log into that PDB, and then perform the `FLUSH SEQUENCE` command.

It is recommended that you use the same schema in each PDB, so that it works with the `GGSCHEMA GLOBALS` parameter file. Here is an example:

```
Environment Information OGG 18.1 Oracle 12c to Oracle 12c Replication,
Integrated Extract, Parallel Replicat
Source: CDB GOLD, PDB CERTMISSN
Target: CDB PLAT, PDB CERTDSQ
Source OGG Configuration
    Container User: C##GGADMIN
```

```
PDB User for Sequences: GGATE
sqlplus / as sysdba
SQL> alter session set container=CERTMISSN;
SQL> create user ggate identified by password default tablespace users
temporary tablespace temp quota unlimited on users container=current;
```

```
Run @sequence
sqlplus / as sysdba
SQL> alter session set container=CERTMISSN;
SQL> @sequence
```

When prompted enter

```
GGATE GLOBALS
GGSCHEMA GGATE
```

```
FLUSH SEQUENCE:
```

```
GGSCI> DBLOGIN USERIDALIAS GGADMIN DOMAIN GOLD_QC_CDB$ROOT
```

```
GGSCI> FLUSH SEQUENCE CERTMISSN.SRCSHEMA1.*
```

Target Oracle GoldenGate Configuration:

```
PDB User: GGATE
Run @sequence
sqlplus / as sysdba
SQL> alter session set container=CERTDSQ;
SQL> @sequence
```

When prompted enter GGATE.

This also applies to the @sequence.sql script, which must also be run at each PDB that you are going to capture from.

Setting Flashback Query

To process certain update records, Extract fetches additional row data from the source database.

Oracle GoldenGate fetches data for the following:

- User-defined types
- Nested tables
- XMLType objects

By default, Oracle GoldenGate uses Flashback Query to fetch the values from the undo (rollback) tablespaces. That way, Oracle GoldenGate can reconstruct a read-consistent row image as of a specific time or SCN to match the redo record.

For best fetch results, configure the source database as follows:

1. Set a sufficient amount of redo retention by setting the Oracle initialization parameters `UNDO_MANAGEMENT` and `UNDO_RETENTION` as follows (in seconds).

```
UNDO_MANAGEMENT=AUTO
```

```
UNDO_RETENTION=86400
```

`UNDO_RETENTION` can be adjusted upward in high-volume environments.

2. Calculate the space that is required in the undo tablespace by using the following formula.

$$undo_space = UNDO_RETENTION * UPS + overhead$$

Where:

- `undo_space` is the number of undo blocks.
- `UNDO_RETENTION` is the value of the `UNDO_RETENTION` parameter (in seconds).
- `UPS` is the number of undo blocks for each second.
- `overhead` is the minimal overhead for metadata (transaction tables, etc.).

Use the system view `V$UNDOSTAT` to estimate `UPS` and `overhead`.

3. For tables that contain LOBs, do one of the following:

- Set the `LOB` storage clause to `RETENTION`. This is the default for tables that are created when `UNDO_MANAGEMENT` is set to `AUTO`.
- If using `PCTVERSION` instead of `RETENTION`, set `PCTVERSION` to an initial value of 25. You can adjust it based on the fetch statistics that are reported with the `STATS EXTRACT` command. If the value of the `STAT_OPER_ROWFETCH_CURRENTBYROWID` or `STAT_OPER_ROWFETCH_CURRENTBYKEY` field in these statistics is high, increase `PCTVERSION` in increments of 10 until the statistics show low values.

4. Grant either of the following privileges to the Oracle GoldenGate Extract user:

```
GRANT FLASHBACK ANY TABLE TO db_user
```

```
GRANT FLASHBACK ON schema.table TO db_user
```

Oracle GoldenGate provides the following parameters to manage fetching.

Parameter or Command	Description
<code>STATS EXTRACT</code> command with <code>REPORTFETCH</code> option	Shows Extract fetch statistics on demand.
<code>STATOPTIONS</code> parameter with <code>REPORTFETCH</code> option	Sets the <code>STATS EXTRACT</code> command so that it always shows fetch statistics.
<code>MAXFETCHSTATEMENTS</code> parameter	Controls the number of open cursors for prepared queries that Extract maintains in the source database, and also for <code>SQLEXEC</code> operations.
<code>MAXFETCHSTATEMENTS</code> parameter	Controls the default fetch behavior of Extract: whether Extract performs a flashback query or fetches the current image from the table.
<code>FETCHOPTIONS</code> parameter with the <code>USELATESTVERSION</code> or <code>NOUSELATESTVERSION</code> option	Handles the failure of an Extract flashback query, such as if the undo retention expired or the structure of a table changed. Extract can fetch the current image from the table or ignore the failure.
<code>REFFETCHEDCOLOPTIONS</code> parameter	Controls the response by Replicat when it processes trail records that include fetched data or column-missing conditions.

Managing Server Resources

Extract interacts with an underlying logmining server in the source database and Replicat interacts with an inbound server in the target database. This section provides guidelines for managing the shared memory consumed by these servers.

The shared memory that is used by the servers comes from the Streams pool portion of the System Global Area (SGA) in the database. Therefore, you must set the database initialization parameter `STREAMS_POOL_SIZE` high enough to keep enough memory available for the number of Extract and Replicat processes that you expect to run in integrated mode. Note that Streams pool is also used by other components of the database (like Oracle Streams, Advanced Queuing, and Datapump export/import), so make certain to take them into account while sizing the Streams pool for Oracle GoldenGate.

By default, one Extract requests the logmining server to run with `MAX_SGA_SIZE` of 1GB. Thus, if you are running three Extracts in the same database instance, you need at least 3 GB of memory allocated to the Streams pool. As a best practice, keep 25 percent of the Streams pool available. For example, if there are 3 Extracts, set `STREAMS_POOL_SIZE` for the database to the following value:

$$3 \text{ GB} * 1.25 = 3.75 \text{ GB}$$

Ensuring Row Uniqueness in Source and Target Tables

Oracle GoldenGate requires a unique row identifier on the source and target tables to locate the correct target rows for replicated updates and deletes.

Unless a `KEYCOLS` clause is used in the `TABLE` or `MAP` statement, Oracle GoldenGate selects a row identifier to use in the following order of priority, depending on the number and type of constraints that were logged (see [Configuring Logging Properties](#)).

1. Primary key if it does not contain any extended (32K) `VARCHAR2/NVARCHAR2` columns. Primary key without invisible columns.
2. Unique key: Unique key without invisible columns.

In the case of a non-integrated Replicat, the selection of the unique key is as follows:

- First unique key alphanumerically with no virtual columns, no UDTs, no function-based columns, no nullable columns, and no extended (32K) `VARCHAR2/NVARCHAR2` columns. To support a key that contains columns that are part of an invisible index, you must use the `ALLOWINVISIBLEINDEXKEYS` parameter in the Oracle GoldenGate `GLOBALS` file.
 - First unique key alphanumerically with no virtual columns, no UDTs, no extended (32K) `VARCHAR2/NVARCHAR2` columns, or no function-based columns, but can include nullable columns. To support a key that contains columns that are part of an invisible index, you must use the `ALLOWINVISIBLEINDEXKEYS` parameter in the Oracle GoldenGate `GLOBALS` file.
3. Not Nullable Unique keys: At least one column from one of the unique keys must be not nullable. This is because `NOALLOWNULLABLEKEYS` is the default.

 **Note:**

`ALLOWNULLABLEKEYS` is not valid for integrated Replicat.

4. Unique key: If the source key and the table key columns do not match and the following criteria is true, then a unique index is chosen that matches with the source table key columns:
 - `KEYCOLS` parameter isn't specified
 - `USEALLKEYCOLUMNS` parameter isn't specified
 - `ALLOWNULLABLEKEYS` parameter isn't specified
 - Source and target key columns don't match
 - Unique index matches to source table key column exists
5. If none of the preceding key types exist (even though there might be other types of keys defined on the table) Oracle GoldenGate constructs a pseudo key of all columns that the database allows to be used in a unique key, excluding virtual columns, UDTs, function-based columns, extended (32K) `VARCHAR2/NVARCHAR2` columns, and any columns that are explicitly excluded from the Oracle GoldenGate configuration by an Oracle GoldenGate user.

Unless otherwise excluded due to the preceding restrictions, invisible columns are allowed in the pseudo key.

**Note:**

If there are other, non-usable keys on a table or if there are no keys at all on the table, Oracle GoldenGate logs an appropriate message to the report file. Constructing a key from all of the columns impedes the performance of Oracle GoldenGate on the source system. On the target, this key causes Replicat to use a larger, less efficient `WHERE` clause.

If a table does not have an appropriate key, or if you prefer the existing key(s) not to be used, you can define a substitute key if the table has columns that always contain unique values. You define this substitute key by including a `KEYCOLS` clause within the Extract `TABLE` parameter and the Replicat `MAP` parameter. The specified key will override any existing primary or unique key that Oracle GoldenGate finds. For more information, see *Parameters and Functions Reference for Oracle GoldenGate*.

Oracle: Supported Data Types, Objects, and Operations for DDL and DML

This chapter contains support information for Oracle GoldenGate on Oracle Database.

Details of Support for Oracle Database Editions

This topic describes the Database Editions from the Oracle Database Product Family supported with the current Oracle GoldenGate release.

Oracle Database Express Edition (XE) is supported for delivery only and does not support any of the integrated features such as integrated Replicat or parallel Replicat in integrated mode.

Oracle Database Standard Edition 2 (SE2) is supported, with the following limitation:

- Extract, integrated Replicat, and parallel Replicat in integrated mode are limited to a single thread.

Oracle Database Enterprise Edition (EE) has full Oracle GoldenGate functionality.

Oracle Database Personal Edition (PE) is supported for delivery only, and does not support any of the integrated features such as integrated or parallel Replicat in integrated mode.

Details of Support for Oracle Data Types and Objects

This topic describes data types, objects and operations that are supported by Oracle GoldenGate.

Within the database, you can use the Dictionary view `DBA_GOLDENGATE_SUPPORT_MODE` to get information about supported objects. There are different types for replication support:

- Support by Capturing from Redo
- Procedural Replication Support

Most data types are supported (`SUPPORT_MODE=FULL`), which implies that Oracle GoldenGate captures the changes out of the redo. In some unique cases, the information cannot be captured, but the information can be fetched with a connection to the database (`SUPPORT_MODE=ID KEY`). Tables supported with `ID KEY` require a connection to the source database or an ADG Standby database for fetching to support those tables. If using downstream Integrated Extract, with `NOUSERID` a customer must specify a `FETCHUSERID` or `FETCHUSERIDALIAS` connection.

Other changes can be replicated with Procedural Replication (`SUPPORT_MODE=PLSQL`) that requires additional parameter setting of Extract. See About Procedural Replication for details. In the unlikely case that there is no native support, no support by fetching and no procedural replication support, there is no Oracle GoldenGate support.

To know more information about capture modes, see [Deciding Which Capture Method to Use](#).

Besides the `DBA_GOLDENGATE_SUPPORT_MODE` at the source database you should check the `DBA_GOLDENGATE_NOT_UNIQUE` dictionary view at the target side. If there are tables without any uniqueness and unbounded data_types (`BAD_COLUMN='Y'`), the table records cannot be uniquely identified and cannot be used for logical replication.

Detailed support information for Oracle data types, objects, and operations starts with [Details of Support for Objects and Operations in Oracle DML](#).

There might be a few cases where replication support exists, but there are limitations of processing such as in case of using `SQLEXEC`. The following table lists these limitations:

Support Datatypes	No Support
NUMBER, BINARY FLOAT, BINARY DOUBLE UROWID	Special cases of: <ul style="list-style-type: none"> • XML types • UDTs • Object tables • Collections or nested tables
DATE and TIMESTAMP	Tables with restricted uniqueness
(N) CHAR, (N) VARCHAR2 LONG, RAW, LONG RAW (N) CLOB, CLOB, BLOB, SECUREFILE, BASICFILE and BFILE	X
XML columns, XMLType	X
UDTs	X
ANYDATA	X

Support Datatypes	No Support
Hierarchy-enabled tables	X
RET Types	X
DICOM	X
SDO_TOPO_GEOMETRY, SDO_GEORASTER	X
Identity columns	X
SDO_RDF_TRIPLE_S	X

**Note:**

SECUREFILE LOBs updated using `DBMS_LOG.FRAGMENT` or SECUREFILE LOBs that are set to `NOLOGGING` are fetched instead of read from the redo.

Supported Capture from Redo:

- NUMBER, BINARY FLOAT, BINARY DOUBLE, and (logical) UROWID
- DATE and TIMESTAMP
- CHAR, VARCHAR2, LONG, NCHAR, and NVARCHAR2
- RAW, LONG RAW, CLOB, NCLOB, BLOB, SECUREFILE, BASICFILE, and BFILE (LOB size limited to 4GB)
- XML columns stored as CLOB, Binary and Object-Relational (OR)
- XMLType columns and XMLType tables stored as XML CLOB, XML Object Relational, and XML Binary
- UDTs (user-defined or abstract data types) on BYTE semantics with source database compatibility 12.0.0.0.0 or higher
- ANYDATA data type with source database compatibility 12.0.0.0.0 or higher
- Hierarchy-enabled tables are managed by the Oracle XML database repository with source database compatibility 12.2.0.0.0 or higher and enabled procedural replication
- REF types with source database compatibility 12.2.0.0.0 or higher
- DICOM with source database compatibility 12.0.0.0.0 or higher
- SDO_TOPO_GEOMETRY or SDO_GEORASTER with source database compatibility 12.2.0.0.0 or higher and enabled procedural replication
- Identity columns with source database compatibility 18.1.0.0.0 or higher
- SDO_RDF_TRIPLE_S with source database compatibility 19.1.0.0.0 or higher

Supported (Fetch from database)**SECUREFILE LOBs**

- Modified with `DBMS_LOB.FRAGMENT_*` procedures
- NOLOGGING LOBs
- Deduplicated LOBs with a source database release less than 12gR2

UDTs that contain following data types:

- `TIMESTAMP WITH TIMEZONE`, `TIMESTAMP WITH LOCAL TIMEZONE`, `TIMESTAMP WITH TIMEZONE` with region ID
- `INTERVAL YEAR TO MONTH`, `INTERVAL DAY TO SECOND`
- `BINARY FLOAT`, `BINARY DOUBLE`
- `BFILE`

Object tables contains the following attributes:

- Nested table
- `SDO_TOP_GEOMETRY`
- `SDO_GEORASTER`

Additional Considerations

- `NUMBER` can be up to the maximum size permitted by Oracle. The support of the range and precision for floating-point numbers depends on the host machine. In general, the precision is accurate to 16 significant digits, but you should review the database documentation to determine the expected approximations. Oracle GoldenGate rounds or truncates values that exceed the supported precision.
- Non-logical `UROWID` columns will be identified by Extract. A warning message is generated in the report file. The column information is not part of the trail record. All other supported datatypes of the record are part of the trail record and are replicated.
- `TIMESTAMP WITH TIME ZONE` as TZR (region ID) for initial loads, `SQLEXEC` or operations where the column can only be fetched from the database. In those cases, the region ID is converted to a time offset by the source database when the column is selected. Replicat applies the timestamp as date and time data into the target database with a time offset value.
- `VARCHAR` expansion from 4K to 32K (extended or long `VARCHAR`)
 - 32K long columns cannot be used as row identifiers:
 - * Columns as part of a key or unique index
 - * Columns in a `KEYCOLS` clause of the `TABLE` or `MAP` parameter.
 - 32K long columns as resolution columns in a CDR (conflict resolution and detection)
 - If an extended `VARCHAR` column is part of unique index or constraint, then direct path inserts to this table may cause Replicat toabend with a warning. Verify that the extended `VARCHAR` caused theabend by checking `ALL_INDEXES` or `ALL_IND_COLUMNS` for a unique index or `ALL_CONS_COLUMNS` or `ALL_CONSTRAINTS` for a unique constraint. Once you determine that an extended `VARCHAR`, you can temporarily drop the index or disable the constraint:
 - * Unique Index: `DROP INDEX index_name;`
 - * Unique Constraint: `ALTER TABLE table_name MODIFY CONSTRAINT constraint_name DISABLE;`
- Oracle GoldenGate does not support the filtering, column mapping, or manipulation of objects larger than 4K.
- `BFILE` column are replicating the locator. The file on the server file system outside of the database and is not replicated.

- Multi-byte character data: The source and target databases must be logically identical in terms of schema definition for the tables and sequences being replicated. Transformation, filtering, and other manipulation cannot be used.
- The character sets between the two databases must be one of the following:
 - Identical on the source and on the target
 - Equivalent, which is not the same character set but containing the same set of characters
 - Target is a superset of the source

Multi-byte data can be used in any semantics: bytes or characters.

- The structure of the UDTs and Abstract Data Types (ADTs) itself must be the same on both the source and target. UDTs can have different source and target schemas. UDTs, including values inside object columns or rows, cannot be used within filtering criteria in `TABLE` or `MAP` statements, or as input or output for the Oracle GoldenGate column-conversion functions, `SQLEXEC`, or other built-in data manipulation tools. Support is only provided for like-to-like Oracle source and targets.

To fully support object tables created using the `CREATE TABLE as SELECT (CTAS)` statement, Integrated Extract must be configured to capture DML from the CTAS statement. Oracle object table can be mapped to a non-Oracle object table in a supported target database.

- XML column type cannot be used for filtering and manipulation. You can map the XML representation of an object to a character column by means of a `COLMAP` clause in a `TABLE` or `MAP` statement.

Oracle recommends the `AL32UTF8` character set as the database character set when working with XML data. This ensures the correct conversion by Oracle GoldenGate from source to target. With DDL replication enabled, Oracle GoldenGate replicates the CTAS statement and allows it to select the data from the underlying target tables. OIDs are preserved if `TRANLOGOPTIONS GETCTASDML` parameter is set. For XMLType tables, the row object IDs must match between source and target.

Non-Supported Oracle Data Types

Oracle GoldenGate does not support the following data types.

- Time offset values outside the range of +12:00 and -12:00..Oracle GoldenGate supports time offset values between +12:00 and -12:00.
- Tables that only contain a single column and that column one of the following:
 - UDT
 - LOB (CLOB, NCLOB, BLOB, BFILE)
 - XMLType column
 - VARCHAR2 (MAX) where the data is greater than 32KB
- Tables with LOB, UDT, XML, or XMLType column without one of the following:
 - Primary Key
 - Scalar columns with a unique constraint or unique index

Table where the combination of all scalar columns do not guarantee uniqueness are unsupported.

- Tables with the following XML characteristics:

- Tables with a primary key constraint made up of XML attributes
- XMLType tables with a primary key based on an object identifier (PKOID).
- XMLType tables, where the row object identifiers (OID) do not match between source and target
- XMLType tables created by an empty CTAS statement.
- XML schema-based XMLType tables and columns where changes are made to the XML schema (XML schemas must be registered on source and target databases with the `dbms_xml` package).
- The maximum length for the entire `SET` value of an update to an XMLType larger than 32K, including the new content plus other operators and XQuery bind values.
- SQL*Loader direct-path insert for XML-Binary and XML-OR.
- Tables with following UDT characteristics:
 - UDTs that contain CFILE or OPAQUE (except of XMLType)
 - UDTs with CHAR and VARCHAR attributes that contain binary or unprintable characters
 - UDTs using the RMTTASK parameter
- UDTs and nested tables with following condition:
 - Nested table UDTs with CHAR, NVARCHAR2 or NCLOB attributes.
 - Nested tables with CLOB, BLOB, extended (32k) VARCHAR2 or RAW attributes in UDTs.
 - Nested table columns/attributes that are part of any other UDT.
- When data in a nested table is updated, the row that contains the nested table must be updated at the same time. Otherwise there is no support.
- When VARRAYS and nested tables are fetched, the entire contents of the column are fetched each time, not just the changes. Otherwise there is no support.
- Object table contains the following attributes:
 - Nested table
 - SDO_TOPO_GEOMETRY
 - SDO_GEORASTER

See additional exclusions in [Details of Support for Oracle Data Types and Objects](#).

Details of Support for Objects and Operations in Oracle DML

This section outlines the Oracle objects and operations that Oracle GoldenGate supports for the capture and replication of DML operations.

Supported Objects and Operations in Oracle DML

Topics:

Multitenant Container Database

Oracle GoldenGate captures from, and delivers to, a **multitenant container database**. See [#unique_173](#).

Oracle GoldenGate does not support application container Root-Child end-to-end replication

Tables, Views, and Materialized Views

The following DML operations are supported for regular tables, index-organized tables, clustered tables, and materialized views:

- INSERT
- UPDATE
- DELETE
- Associated transaction control operations

Tip:

You can use the `DBA_GOLDENGATE_SUPPORT_MODE` data dictionary view to display information about the level of Oracle GoldenGate capture process support for the tables in your database. The `PLSQL` value of `DBA_GOLDENGATE_SUPPORT_MODE` indicates that the table is supported natively, but requires procedural supplemental logging.

Besides the `DBA_GOLDENGATE_SUPPORT_MODE` at the source database, you should check the `DBA_GOLDENGATE_NOT_UNIQUE` dictionary view at the target side. If there are tables without any uniqueness and unbounded data types (`BAD_COLUMN='Y'`), the table records cannot be uniquely identified and cannot be used for logical replication.

For more information, see the `DBA_GOLDENGATE_SUPPORT_MODE`. If you need to display all tables that have no primary and no non-null unique indexes, you can use the `DBA_GOLDENGATE_NOT_UNIQUE`. For more information, see `DBA_GOLDENGATE_NOT_UNIQUE`.

Limitations of Support for Regular Tables

These limitations apply to Extract.

- Oracle GoldenGate supports tables that contain any number of rows.
- A row can be up to 4 MB in length. If Oracle GoldenGate is configured to include both the before and after image of a column in its processing scope, the 4 MB maximum length applies to the total length of the full before image plus the length of the after image. For example, if there are `UPDATE` operations on columns that are being used as a row identifier, the before and after images are processed and cannot exceed 4 MB in total. Before and after images are also required for columns that are not row identifiers but are used as comparison columns in conflict detection and resolution (CDR). Character columns that allow for more than 4 KB of data, such as a `CLOB`, only have the first 4 KB of data stored in-row and contribute to the 4MB maximum row length. Binary columns that allow for more than 4kb of data, such as a `BLOB` the first 8 KB of data is stored in-row and contributes to the 4MB maximum row length.
- Oracle GoldenGate supports the maximum number of columns per table that is supported by the database.
- Oracle GoldenGate supports the maximum column size that is supported by the database.
- Oracle GoldenGate supports tables that contain only one column, except when the column contains one of the following data types:
 - LOB

- LONG
- LONG VARCHAR
- Nested table
- User Defined Type (UDT)
- VARRAY
- XMLType
- Set `DBOPTIONS ALLOWUNUSEDCOLUMN` before you replicate from and to tables with unused columns.
- Oracle GoldenGate supports tables with these partitioning attributes:
 - Range partitioning
 - Hash Partitioning Interval Partitioning
 - Composite Partitioning
 - Virtual Column-Based Partitioning
 - Reference Partitioning
 - List Partitioning
- Oracle GoldenGate supports tables with virtual columns, but does not capture change data for these columns or apply change data to them: The database does not write virtual columns to the transaction log, and the Oracle Database does not permit DML on virtual columns. For the same reason, initial load data cannot be applied to a virtual column. You can map the data from virtual columns to non-virtual target columns.
- Oracle GoldenGate will not consider unique/index with virtual columns.
- Oracle GoldenGate supports replication to and from Oracle Exadata. To support Exadata Hybrid Columnar Compression, Extract must operate in integrated capture mode. To support Exadata Hybrid Columnar Compression, the source database compatibility must be set to 11.2.0.0.0 or higher.
- Oracle GoldenGate supports Transparent Data Encryption (TDE).
 - Extract supports TDE column encryption and TDE table space encryption without setup requirements in integrated capture mode.
- Oracle GoldenGate supports `TRUNCATE` statements as part of its DDL replication support, or as standalone functionality that is independent of the DDL support.
- Oracle GoldenGate supports the capture of direct-load `INSERT`, with the exception of SQL*Loader direct-path insert for XML Binary and XML Object Relational. Supplemental logging must be enabled, and the database must be in archive log mode. The following direct-load methods are supported.
 - `/*+ APPEND */ hint`
 - `/*+ PARALLEL */ hint` (Not supported for RAC in classic capture mode)
 - `SQLLDR with DIRECT=TRUE`
- Oracle GoldenGate fully supports capture from compressed objects when Extract is in integrated capture mode. Extract in classic capture mode does not support compressed objects.
- Oracle GoldenGate supports XA and PDML distributed transactions in integrated capture mode. Extract in classic capture mode does not support PDML or XA on RAC.

- Oracle GoldenGate supports DML operations on tables with `FLASHBACK ARCHIVE` enabled. However, Oracle GoldenGate does not support DDL that creates tables with the `FLASHBACK ARCHIVE` clause or DDL that creates, alters, or deletes the flashback data archive itself.

Limitations of Support for Views

These limitations apply to Extract.

- Oracle GoldenGate supports capture from a view when Extract is in initial-load mode (capturing directly from the source view, not the redo log).
- Oracle GoldenGate does not capture change data from a view, but it supports capture from the underlying tables of a view.

Limitations of Support for Materialized Views

Materialized views are supported by Extract with the following limitations.

- Materialized views created `WITH ROWID` are not supported.
- The materialized view log can be created `WITH ROWID`.
- The source table must have a primary key.
- Truncates of materialized views are not supported. You can use a `DELETE FROM` statement.
- DML (but not DDL) from a full refresh of a materialized view is supported. If DDL support for this feature is required, open an Oracle GoldenGate support case.
- For Replicat the `Create MV` command must include the `FOR UPDATE` clause
- Either materialized views can be replicated or the underlying base table(s), but not both.

Limitations of Support for Clustered Tables

Indexed clusters are supported by Extract while hash clusters are not supported.

System Partitioning

System partitioning is an Oracle database feature that allows a table to be created with named partitions. A system partitioned table is not maintained by the database. Each DML must specify the partition where the row is to reside. Extract and all modes of Replicat support system partitioning. Each trail file record header pertaining to a system partitioned table includes the partition name.

See `PARTITION` | `PARTITIONEXCLUDE` in the *Parameters and Functions Reference for Oracle GoldenGate*.

Sequences and Identity Columns

- Identity columns are supported from Oracle database 18c onward and requires Extract, Parallel Replicat in Integrated mode, or Integrated Replicat.
- Oracle GoldenGate supports the replication of sequence values and identity columns in a unidirectional and active-passive high-availability configuration.
- Oracle GoldenGate ensures that the target sequence values will always be higher than those of the source (or equal to them, if the cache is zero).

Limitations of Support for Sequences

These limitations apply to Extract.

- Oracle GoldenGate does not support the replication of sequence values in an active-active bi-directional configuration.
- 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.
- Tables with default sequence columns are excluded from replication for Extract.

Non-supported Objects and Operations in Oracle DML

The following are additional Oracle objects or operations that are not supported by Extract:

- `REF` are supported natively for compatibility with Oracle Database 12.2 and higher, but not primary-key based `REFs` (`PKREFs`)
- Sequence values in an active-active bi-directional configuration
- Database Replay
- Tables created as `EXTERNAL`

DML Auto Capture

Oracle GoldenGate supports the following DML operations with auto capture mode:

- `TABLEEXCLUSION` parameter is supported.
- `TABLE` parameter is supported.
- Extract writes the table DML records delivered by the database for auto capture to trail file.

Details of Support for Objects and Operations in Oracle DDL

This topic outlines the Oracle objects and operation types that Oracle GoldenGate supports for the capture and replication of DDL operations.

Trigger-based capture is required for Oracle releases that are earlier than version 11.2.0.4. If Extract will run in integrated mode against a version 11.2.0.4 or later of Oracle Database, then the DDL trigger and supporting objects are not required.

Supported Objects and Operations in Oracle DDL

When the source database is Oracle 11.2.0.4 or later and Extract operates in integrated mode, DDL capture support is integrated into the database logmining server and does not require the use of a DDL trigger. You must set the database parameter compatibility to 11.2.0.4.0. Extract supports DDL that includes password-based column encryption, such as:

- `CREATE TABLE t1 (a number, b varchar2(32) ENCRYPT IDENTIFIED BY my_password);`
- `ALTER TABLE t1 ADD COLUMN c varchar2(64) ENCRYPT IDENTIFIED BY my_password;`



Note:

Password-based column encryption in DDL is not supported in classic capture mode.

The following additional statements apply to Extract with respect to DDL support.

- All Oracle GoldenGate topology configurations are supported for Oracle DDL replication.
- Active-active (bi-directional) replication of Oracle DDL is supported between two (and only two) databases that contain identical metadata.
- Oracle GoldenGate supports DDL on the following objects:
 - clusters
 - directories
 - functions
 - indexes
 - packages
 - procedure
 - tables
 - tablespaces
 - roles
 - sequences
 - synonyms
 - triggers
 - types
 - views
 - materialized views
 - users
 - invisible columns
- Oracle Edition-Based Redefinition (EBR) database replication of Oracle DDL is supported for integrated Extract for the following Oracle Database objects:
 - functions
 - library
 - packages (specification and body)
 - procedure
 - synonyms
 - types (specification and body)
 - views

EBR does not support use of DDL triggers.

- Oracle GoldenGate supports DDL operations of up to 4 MB in size. Oracle GoldenGate measures the size of DDL statement in bytes, not in characters. This size limitation includes packages, procedures, and functions. The actual size limit of the DDL support is approximate, because the size not only includes the statement text, but also Oracle GoldenGate maintenance overhead that depends on the length of the object name, the DDL type, and other characteristics of keeping a DDL record internally.

- Oracle GoldenGate supports Global Temporary Tables (GTT) DDL operations to be visible to Extract so that they can be replicated. You must set the `DDLOPTIONS` parameter to enable this operation because it is not set by default.
- Oracle GoldenGate supports Integrated Dictionary for use with `NOUSERID` and `TRANLOGOPTIONS GETCTASDML`. This means that Extract will be obtaining object metadata from the LogMiner dictionary instead of the DDL trigger and without querying the dictionary objects. Oracle GoldenGate uses Integrated Dictionary automatically when the source database compatibility parameter is greater than or equal to 11.2.0.4 and Integrated Extract is used.

The Integrated Dictionary feature is *not* supported with classic Extract.

When using Integrated Dictionary and trail format in the Oracle GoldenGate release 12.2.x, Integrated Capture requires the Logminer patch to be applied on the mining database if the Oracle Database release is earlier than 12.1.0.2.

- Oracle GoldenGate supports replication of invisible columns in Integrated Capture mode. Trail format release 12.2 is required. Replicat must specify the `MAPINVISIBLECOLUMNS` parameter or explicitly map to invisible columns in the `COLMAP` clause of the `MAP` parameter.

If `SOURCEDEFS` or `TARGETDEFS` is used, the metadata format of a definition file for Oracle tables must be compatible with the trail format. Metadata format 12.2 is compatible with trail format 12.2, and metadata format earlier than 12.2 is compatible with trail format earlier than 12.2. To specify the metadata format of a definition file, use the `FORMAT` `RELEASE` option of the `DEFSFILE` parameter when the definition file is generated in `DEFGEN`.

- DDL statements to create a namespace context (`CREATE CONTEXT`) are captured by Extract and applied by Replicat.
- Extract in pump mode supports the following DDL options:
 - `DDL INCLUDE ALL`
 - `DDL EXCLUDE ALL`
 - `DDL EXCLUDE OBJNAME`

The `SOURCECATALOG` and `ALLCATALOG` option of `DDL EXCLUDE` is also supported.

If no DDL parameter is specified, then all DDLs are written to trail. If `DDL EXCLUDE OBJNAME` is specified and the object owner is does not match an exclusion rule, then it is written to the trail.

Non-supported Objects and Operations in Oracle DDL

These statements apply to integrated and classic capture modes.

Excluded Objects

The following names or name prefixes are considered Oracle-reserved and must be excluded from the Oracle GoldenGate DDL configuration. Oracle GoldenGate will ignore objects that contain these names.

Excluded schemas:

```
"ANONYMOUS", // HTTP access to XDB
"APPQOSSYS", // QOS system user
"AUDSYS", // audit super user
"BI", // Business Intelligence
"CTXSYS", // Text
"DBSNMP", // SNMP agent for OEM
```

```
"DIP", // Directory Integration Platform
"DMSYS", // Data Mining
"DVF", // Database Vault
"DVSYS", // Database Vault
"EXDSYS", // External ODCI System User
"EXFSYS", // Expression Filter
"GSMADMIN_INTERNAL", // Global Service Manager
"GSMCATUSER", // Global Service Manager
"GSMUSER", // Global Service Manager
"LBACSYS", // Label Security
"MDSYS", // Spatial
"MGMT_VIEW", // OEM Database Control
"MDDATA",
"MTSSYS", // MS Transaction Server
"ODM", // Data Mining
"ODM_MTR", // Data Mining Repository
"OJVMSYS", // Java Policy SRO Schema
"OLAPSYS", // OLAP catalogs
"ORACLE_OCM", // Oracle Configuration Manager User
"ORDDATA", // Intermedia
"ORDPLUGINS", // Intermedia
"ORDSYS", // Intermedia
"OUTLN", // Outlines (Plan Stability)
"SI_INFORMTN_SCHEMA", // SQL/MM Still Image
"SPATIAL_CSW_ADMIN", // Spatial Catalog Services for Web
"SPATIAL_CSW_ADMIN_USR",
"SPATIAL_WFS_ADMIN", // Spatial Web Feature Service
"SPATIAL_WFS_ADMIN_USR",
"SYS",
"SYSBACKUP",
"SYSDG",
"SYSKM",
"SYSMAN", // Administrator OEM
"SYSTEM",
"TSMSYS", // Transparent Session Migration
"WKPROXY", // Ultrasearch
"WKSYS", // Ultrasearch
"WK_TEST",
"WMSYS", // Workspace Manager
"XDB", // XML DB
"XS$NULL",
"XTISYS", // Time Index
```

Special schemas:

```
"AURORA$JIS$UTILITY$", // JSERV
"AURORA$ORB$UNAUTHENTICATED", // JSERV
"DSSYS", // Dynamic Services Secured Web Service
"OSE$HTTP$ADMIN", // JSERV
"PERFSTAT", // STATSPACK
"REPADMIN",
"TRACESVR" // Trace server for OEM
```

Excluded tables (the * wildcard indicates any schema or any character):

```
"*.AQ$*", // advanced queues
"*.DR$*$*", // oracle text
"*.M* *$$", // Spatial index
"*.MLOG$*", // materialized views
"*.OGGQT$*",
"*.OGG$*", // AQ OGG queue table
"*.ET$*", // Data Pump external tables
"*.RUPD$*", // materialized views
```

```

"*.SYS_C*", // constraints
"*.MDR*_*$", // Spatial Sequence and Table
"*.SYS_IMPORT_TABLE*",
"*.CMP*$*", // space management, rdbms >= 12.1
"*.DBMS_TABCOMP_TEMP_*", // space management, rdbms < 12.1
"*.MDXT_*$*" // Spatial extended statistics tables

```

Other Non-supported DDL

Oracle GoldenGate does not support the following:

- DDL on nested tables.
- DDL on identity columns.
- ALTER DATABASE and ALTER SYSTEM (these are not considered to be DDL) Using dictionary, you can replicate ALTER DATABASE DEFAULT EDITION and ALTER PLUGGABLE DATABASE DEFAULT EDITION. All other ALTER [PLUGABLE] DATABASE commands are ignored.
- DDL on a standby database.
- Database link DDL.
- DDL that creates tables with the FLASHBACK ARCHIVE clause and DDL that creates, alters, or deletes the flashback data archive itself. DML on tables with FLASHBACK ARCHIVE is supported.
- Some DDL will generate system generated object names. The names of system generated objects may not always be the same between two different databases. So, DDL operations on objects with system generated names should only be done if the name is exactly the same on the target.

Prepare Oracle GoldenGate

Learn about the prerequisite tasks to be completed before beginning to add Extract and Replicat processes for Oracle GoldenGate deployments.

Oracle GoldenGate Users

A user needs to be created in the source database, if you are using Oracle GoldenGate DDL support. This user performs maintenance on the Oracle GoldenGate database objects that support DDL capture.

A user is required in either the source or target database for the DEFGEN utility. The location depends on where the data definition file is being generated. This user performs local metadata queries to build a data-definitions file that supplies the metadata to remote Oracle GoldenGate instances.

Configure Secure Database Connections from Oracle GoldenGate

To specify a database connection string in a secure manner while configuring Oracle GoldenGate connections to any of the supported databases, the following options are available:

- Include the USERIDALIAS option in the Extract and Replicat parameter files
- Set up a connection using TCP or Bequeath protocols

! Important:

For Oracle database, it is recommended that you use the TCP or Bequeath protocols with Oracle GoldenGate to be able to use features such as efficient DDL notification. Avoid using the IPC protocol as there are intermittent issues with using this protocol. For details, see [Table DDL Change Notification](#) in the *Oracle Database Development Guide*

Security Options for Specifying the Connection String in the Extract and Replicat Parameter Files

The following are the security options for specifying the connection string in the Extract or Replicat parameter file.

Credential store method:

```
USERIDALIAS ggeast
```

In the case of `USERIDALIAS`, the alias `ggeast` is stored in the Oracle GoldenGate credential store with the actual connection string. The following example uses the `INFO CREDENTIALSTORE` command to display the details of the credentials configured in Oracle GoldenGate:

```
INFO CREDENTIALSTORE DOMAIN OracleGoldenGate
```

Output:

```
Domain: OracleGoldenGate
Alias: ggeast
Uuserid: ggadmin@dc1.example.com:1521/DBEAST.example.com
```

Setting up a Bequeath connection

Valid for Oracle database.

Oracle GoldenGate can connect to a database instance without using the network listener if a Bequeath connect descriptor is added in the `tnsnames.ora`.

The following example shows the configuration for connecting to a database using Bequeath connect descriptor:

```
dbbeq = (DESCRIPTION=
  (ADDRESS= (PROTOCOL=beq)
    (ENVS='ORACLE_SID=sales,ORACLE_HOME=/app/db_home/oracle,LD_LIBRARY_PATH=/app/db_home/oracle/lib')
    (PROGRAM=/app/db_home/oracle/bin/oracle)
    (ARGV0=oraclesales)
    (ARGS=' (DESCRIPTION= (LOCAL=YES) (ADDRESS= (PROTOCOL=beq) ) ) )
    (CONNECT_DATA= (SID=sales) ) )
```

In this example:

`/app/db_home` is the target Oracle database installation directory

`sales` is the database service name

The `ORACLE_SID`, `ORACLE_HOME`, and `LD_LIBRARY_PATH` in the `ENVS` parameter refers to the target.



Note:

Make sure that there is no white space between these environment variable settings.

Setting up a TCP connection

For Oracle database, you can configure connect description in the `tnsnames.ora` file for setting up a TCP connection and save it in the credentials store in Oracle GoldenGate. The following example shows the `tnsnames.ora` file with the TCP connect descriptor:

```
##tnsnames.ora file sample for database host DBEAST
cdb23_root  = (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=DBEAST) (PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=rdbms.oracle.com)))
cdb23_pdb0  = (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=DBEAST) (PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=cdb1_pdb0.rdbms.oracle.com)))
cdb23_pdbeast = (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=DBEAST) (PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=cdb1_pdbeast.rdbms.oracle.com)))
cdb23_pdbwest = (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=DBEAST) (PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=cdb1_pdbwest.rdbms.oracle.com)))
```

To configure additional security options using `sqlnet.ora`, see [Connecting to a Database Using Strong Authentication](#)

Assigning Credentials to Oracle GoldenGate

The Oracle GoldenGate processes require one or more database credentials with the correct database privileges for the database version, database configuration, and Oracle GoldenGate features that you are using.

Create users for the source and target database instances, each one dedicated to Oracle GoldenGate. The assigned user can be the same user for all the Oracle GoldenGate processes that must connect to a source or target Oracle Database.

See `ALTER CREDENTIALSTORE` command usage to manage credentials in a credential store for Oracle GoldenGate users.

Securing the Oracle GoldenGate Credentials

To preserve the security of your data, and to monitor Oracle GoldenGate processing accurately, do not permit other users, applications, or processes to log on as, or operate as, an Oracle GoldenGate database user.

Oracle GoldenGate provides different options for securing the log-in credentials assigned to Oracle GoldenGate processes. The recommended option is to use a credential store. You can create one credential store and store it in a shared location where all installations of Oracle GoldenGate can access it, or you can create a separate one on each system where Oracle GoldenGate is installed.

The credential store stores the user name and password for each of the assigned Oracle GoldenGate users. A user ID is associated with one or more aliases, and it is the alias that is supplied in commands and parameter files, not the actual user name or password. The

credential file can be partitioned into domains, allowing a standard set of aliases to be used for the processes, while allowing the administrator on each system to manage credentials locally.

See *Creating and Populating the Credential Store in Oracle GoldenGate Security Guide* for more information about creating a credential store and adding user credentials.

Add and Alter Database Credentials

To create and run Extract and Replicat processes, you need to set up database credentials.

1. Launch the Administration Server interface and log in.
2. Click **Configuration** from the **Application Navigation** pane.
3. Click the **+** sign next to Credentials, and set up your new credential alias, then click **Submit**.
4. Click the Login icon to verify that the new alias can correctly log in to the database.

If an error occurs, click the **Alter Credential** icon to correct the credential information, and then test the log in.

You can edit existing credentials to change the user name and password. Delete a credential by clicking the trash icon.

When you successfully log into your database, you can add and manage checkpoint tables, transaction information, and heartbeat tables. All of the tables can be searched using the various search fields. As you type, the table is filtered and you can use the search button with the search text.

Before Adding Extract and Replicat Processes

Learn about the prerequisite configurations required before creating Extract and Replicat processes for an Oracle GoldenGate deployment.

Add TRANDATA

Valid for Db2 i, Db2 LUW, Db2 z/OS, Oracle, PostgreSQL, SQL Server, and Sybase.

Depending on the source database, supplemental logging must be enabled to capture DML operations and can be enabled through the **Trandata** menu of a database connection in the web interface, or in the Admin Client by issuing `ADD TRANDATA` or `ADD SCHEMATRANDATA` (for Oracle only).

Adding `TRANDATA` is not required on a source database for an initial load Extract. However, if both initial load Extract and change data capture (CDC) Extract will be used in conjunction, for an online instantiation, then `TRANDATA` should be added prior to starting the initial load Extract.

Enable TRANDATA or SCHEMATRANDATA for Oracle Database

This can be done at the table, schema, or global (database) level.

To enable supplemental logging, connect to the database from the **DB Connections** page, select the **Trandata** menu, then perform the following steps:

1. Select the Table or Schema option as required and click plus sign to add.
2. Enter the name of the table for which you need to set up supplemental logging. Make sure to enter the full table name with schema name, such as, `HR.EMP`. You can also use wildcard instead of specific table name.

3. Click **Submit**.

You can skip `ADD TRANDATA` in case of initial load without CDC.

You can also use the commands `ADD TRANDATA` and `ADD SCHEMATRANDATA` for setting up trandata and schema level trandata. For details, see [ADD TRANDATA](#) and [ADD SCHEMATRANDATA](#) in *Command Line Interface Reference for Oracle GoldenGate*.



Note:

Before you run the `ADD TRANDATA` command, you need to first connect to the database where the Extract will be added, using the `DBLOGIN` command. In addition, run the `ADD TRANDATA` or `ADD SCHEMATRANDATA` commands before adding the Extract.

Add a Checkpoint Table

A checkpoint table is required for all non-parallel Replicats and must be created in the database prior to adding a Replicat. You can view the checkpoint table within the checkpoint section. To add a checkpoint table, connect to the target database from the **DB Connections** page, select **Checkpoint**, then follow the steps below.

1. Click the **plus sign** to enable adding a checkpoint table.
2. Add the checkpoint table name in the format

```
table.checkpoint_table_name
```

.

3. Click **Submit**. After the checkpoint is created, you'll be able to see in the list of checkpoint tables.

To perform this task from the command line, see [ADD CHECKPOINTTABLE](#) in the *Command Line Interface Reference for Oracle GoldenGate*.

Add Heartbeat Table

Heartbeat tables are used to monitor lag throughout the data replication cycle. Automatic heartbeats are sent from each source database into the replication streams, by updating the records in a heartbeat seed table and a heartbeat table, and constructing a heartbeat history record.

Each process in the replication stream updates the heartbeat record with tracking information which is then updated in the heartbeat table of the target database. These heartbeat records are inserted or updated into the heartbeat table at the target databases.



Note:

Creating the heartbeat table is optional but is recommended.

To add a heartbeat table, connect to each source and target database from the DB Connections page, select the Heartbeat menu, then perform the following steps:

1. Click the plus (+) sign next to add a heartbeat table.

2. Accept the default settings or modify the available values as needed.

 **Note:**

For databases that have an option for **Target Only**, select this option if that database is only going to be used as a target database in the replication stream, to avoid creating unnecessary jobs that would be associated with a source database.

3. Click **Submit**.

To perform this task from the command line and review important database specific limitations,, see `ADD HEARTBEATTABLE` in *Command Line Interface Reference for Oracle GoldenGate*.

The following steps describe the commands to set up the heartbeat table.

1. Launch the Admin Client from the command line.
2. Connect to the deployment from the Admin Client.

```
CONNECT https://remotehost:srvmgrport DEPLOYMENT deployment_name AS  
deployment_user PASSWORD deployment_password
```

Here's an example:

```
CONNECT https://remotehost:16000 DEPLOYMENT ggdep_postgres AS ggadmin  
PASSWORD P@ssWord
```

3. Connect to the source and target databases using the `DBLOGIN USERIDALIAS` command. The following example shows the connection to the source database with credential alias `ggeast`:

```
(https://remotehost:16000 ggdep_postgres)> DBLOGIN USERIDALIAS ggeast
```

4. Add the heartbeat table:

```
(https://remotehost:16000 ggdep_postgres)> ADD HEARTBEATTABLE
```

Optionally, for a target only database, one that is used for unidirectional replication only, you can include the `TARGETONLY` option which will not create a heartbeat record update function.

See [ADD HEARTBEATTABLE](#) for details about command options.

5

Extract

Learn about different types of Extract and how to add and manage Extracts.

Quick Tour of the Administration Service Overview Page

When you click the Administrator Server link on the Service Manager home page, the login page for the Administration Server is displayed. After logging in, you can configure Extract and Replicat processes from this Web UI.

The Administration Server home page is used to add Extracts and Replicats. The table on the home page displays the severity of critical events. You can also use the left-navigation pane to access various configuration details, a list of severity issues with their diagnosis, and a list of administrators.

Now, that you have an overview of the Administration Server home page, let's understand some of the key actions that you can perform from this page.

Action	Description
View the home page in tabular format	Use the Table Layout swivel to turn the tabular format on and off.
View Extracts and Replicats	The statistical representation the home page displays current state of Extracts and Replicats (Starting, Running, Stopped, Abended, Killed)
Add an Extract	See How to Add an Extract for a Deployment
Create a Replicat	See How to Add a Replicat
Stop and start Extracts	Using Extract Actions
Stop and start Replicats	See Using Replicat Actions
View and search critical events	Monitor severity of events using the Critical Events table and also search for specific events, if required.

About Extract

The Extract process is configured to run against the source database, capturing data generated in the true source database located somewhere else. This process is the extraction or the data capture mechanism of Oracle GoldenGate.

You can configure an Extract for the following use cases:

- **Initial Loads:** When you set up Oracle GoldenGate for initial loads, the Extract process captures the current, static set of data directly from the source objects. This configuration of Extract process uses source tables as the source to capture data. See [Add Initial Load Extract Using the Admin Client](#).
- **Online Extract (Change Synchronization):** When you set up Oracle GoldenGate to keep the source data synchronized with another set of data, the Extract process captures the

DML and DDL operations performed on the configured objects after the initial synchronization has taken place. Extracts can run locally (upstream) on the same server as the database or on another server using the downstream integrated Extract (in case of Oracle database) for reduced overhead. It stores these operations until it receives commit records or rollbacks for the transactions that contain them. If it receives a rollback, it discards the operations for that transaction. If it receives a commit, it persists the transaction to disk in a series of files called a trail, where it is queued for propagation to the target system. All the operations in each transaction are written to the trail as a sequentially organized transaction unit and are in the order in which they were committed to the database (commit sequence order). This design ensures both speed and data integrity. This configuration of the Extract process uses database recovery logs or transaction logs as the data source. While capturing from the logs, the actual method varies depending on the database type. An example of this source type is the Oracle database redo logs, which are used for supplemental logging.

**Note:**

Extract ignores operations on objects that are not in the Extract configuration, even though a transaction may also include operations on objects that are in the Extract configuration.

For a remote deployment, the source database and Oracle GoldenGate are installed on separate servers. Remote deployments are the only option available for supporting cloud databases, such as Azure for PostgreSQL or Amazon Aurora PostgreSQL.

For remote deployments, operating system endianness between the database server and Oracle GoldenGate server need to be the same.

Server time and time zones of the Oracle GoldenGate server should be synchronized with that of the database server. If this is not possible, then positioning of an Extract when creating or altering one will need to be done by LSN.

In remote capture use cases, using `SQLEXEC` may introduce additional latency, as the `SQLEXEC` operation must be done serially for each record that the Extract processes. If special filtering that would require a `SQLEXEC` is done by a remote hub Extract and the performance impact is too severe, it may become necessary to move the Extract process closer to the source database.

With remote deployments, low network latency is important, and it is recommended that the network latency between the Oracle GoldenGate server and the source database server be less than 1 millisecond.

Add an Extract

Set up database credentials to create and run Extracts using the steps in [Add and Alter Database Credentials](#).

1. Log in to the Administration Server using the Oracle GoldenGate user credentials.
2. From the Overview page of the Administration Server, click the + sign next to Extract.
3. Choose the type of Extract to create and click **Next**. The types of Extract are:
 - Integrated Extract
 - Classic Extract

- Initial Load Extract

 **Note:**

An Initial Load Extract cannot be started from a secure deployment. You can only start it in a non-secure deployment.

- Enter and select the required information, which is designated with an asterisk (*). For all Extracts the Process Name, Credential Domain, and Credential Alias are required. A description is optional. The **Create new credential** option is common to all Extracts.


You can configure the following additional required and optional details based on the type of Extract you selected to create:

Options	Description Description	Database
Basic Information		
Process Name	Name of the Extract process. The name of the Extract process can be up to 8 characters.	All databases
Description	Description of the Extract process being created.	All databases
Intent	Describes the purpose of creating the Extract. The default option is Unidirectional. Other options are High Availability, Disaster Recovery, N-Way, which are informational only.	All databases
Begin	Used to set the beginning location in the redo or transaction log from which the Extract will start to capture data. Available options are Now, Custom Time, CSN or Position in Log, and EOF depending on the supported database.	All databases
Trail Name	A two character trail name.	All databases
Trail Subdirectory, Size, Sequence, and Offset	You can further configure the trail details.	All databases
Remote	Enable this option if the Extract trail is remote. For Oracle databases, enable this option if the Extract trail is to be written directly to a remote Oracle GoldenGate Classic installation. For MySQL, setting this option enables the <code>TRANLOGOPTIONS ALTLOGDEST REMOTE</code> parameter to support a remote Extract, and is not related to trails.	Oracle, MySQL
Registration Information		

Options	Description Description	Database
CSN	Commit Sequence Number (CSN) value	Oracle
Share	Choose the method to share the LogMiner data dictionary. Options are: <ul style="list-style-type: none"> Automatic: This option allows the system to choose the method for sharing the dictionary . None: Choosing this option, will not allow the dictionary to be shared. Extract: Choose this option to allow sharing the logminer dictionary for specific Extract. 	Oracle
Optimized	Enable this option to optimize the Extract registration.	Oracle
Downstream Capture	Enable this option to set up a downstream Extract for log mining.	Oracle
Register Only	Use this option to just register the Extract and not add the Extract. The registration creates the replication slot when you register the Extract or use the Register Only option.	PostgreSQL
Source Database Credential		
Create new credential	If you haven't set up your database login credentials, you can create and save the database login credentials from here.	All
Credential Domain	Create a domain for the database.	All
Credential Alias	Specify a credential for the database login.	All
User ID	Specify a user name for logging into the database.	All
Password, Verify Password	Enter the password used to login to the database and reenter the password to verify.	All
Credential 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.	All databases

Options	Description Description	Database
Credential 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.	All databases
Downstream Mining		
Mining Credential Domain	Domain name of the downstream mining database.	Oracle
Mining Credential Alias	Alias for the mining downstream database.	Oracle
No UserID	Enable this option if there is no source database connection. Selecting this option enables the ADG fetch options.	Oracle
ADG Fetch Credential Domain	Domain name for the ADG fetch database.	Oracle
ADG Fetch Credential Alias	Domain alias for the ADG fetch database.	Oracle

You must enter the options for Managed Processes while creating all types of Extract processes. The following table provides these options:

Option	Description
Profile Name	Provides the name of the autostart and autorestart profile. You can select the default or custom options. If you have already created a profile, then you can select that profile also. If you select the Custom option, then you can set up a new profile from this section itself.
Critical to deployment health	(Oracle only) Enable this option if the profile is critical for the deployment health. This option can be enabled for High Availability environments.
<div>  Note: This option only appears while creating the Extract or Replicat and not when you set up the managed processes in the Profiles page. </div>	
Auto Start	Enables autostart for the process.

Option	Description
Startup Delay	Time to wait in seconds before starting the process
Auto Restart	Configures how to restart the process if it terminates
Max Retries	Specify the maximum number of retries to try to start the process
Retry Delay	Delay time in trying to start the process
Retries Window	The duration interval to try to start the process
Restart on Failure only	If true the task is only restarted if it fails
Disable Task After Retries Exhausted	If true then the task is disabled after exhausting all attempts to restart the process.

5. Click **Next**.
6. You can edit the parameter file in the text area to list the table details that you are interested in capturing. For example, `table source.table1;`

You can select **Register Extract in the background** to register the Extract in the background asynchronously.
7. You can select Register Extract in the background to register the Extract in the background asynchronously.
8. Click **Create and Run** to create and start the Extract. If you select **Create**, the Extract is created but you need to start it using the Extract drop-down on the Overview page.

You are returned to the Overview page of the Administration Server. Select the Action list if you want to look at the Extract details such as process information, checkpoint, statistics, parameters, and report.

Using Extract Actions

Once you create an Extract, you can monitor various details associated with the Extract from the Administration Server home page.

You can change the status of the Extract process using the Action button to:

Action	Result
Details	<p>Displays the following tabs:</p> <ul style="list-style-type: none">• Process Information: The status of the selected process including the type, credentials, and trail.• Checkpoint: The checkpoint log name, path, timestamp, sequence, and offset value. You can monitor the input details, such as when starting, at recovery, and the current state. The checkpoint output values display the current checkpoint details.• Statistics: The active replication maps along with replication statistics based on the process type. You sort the list to view the entire statistical data, daily, or hourly basis.• Parameters: The parameters configured when the process was added. You can edit the parameters by clicking the pencil icon. Make sure that you apply your changes.• Report: A detailed report of the process including parameter settings and a log of the transactions. You could copy the report text and save it to a file so that you can share or archive it.
Start/Stop	The Extract starts or stops immediately.
Start/Stop (in the background)	The Extract is started or stopped using a background process.
Start with Options	Allows you to change the Extract CSN options, then starts the Extract.
Alter	This option is available only when the Extract is stopped. Allows you to change when the Extract begins, the description, and the intent. It does not start the Extract.
Delete	This option displays only when the Extract is stopped. Deletes the Extract if you confirm the deletion.

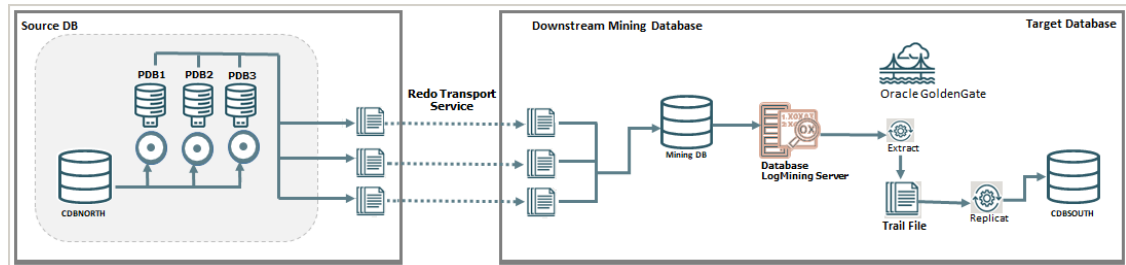
When you change the status, the list options change accordingly. As status are changing, the icons change to indicate the current and final status. The events are added to the Critical Events table. Additionally, progress pop-up notifications appear at the bottom of the page.

Downstream Extract for Downstream Database Mining

Learn about configuring downstream database mining with Oracle GoldenGate Extract and cascaded downstream database mining using Active Data Guard (ADG).

Configure Extract for a Downstream Deployment

A downstream Oracle GoldenGate deployment allows you to offload the source database redo logs to a downstream mining database. A downstream mining database can accept both archived logs and online redo logs from a source database.



This workflow shows the source multitenant container database (CDBNORTH) offloading redo logs to the downstream database (CDBSOUTH) through the logmining server.



Note:

Configuring Extract for a downstream deployment is only applicable to Oracle database.

important for any Data Guard or downstream environment is the setup of the Redo Transport which with the parameter settings of

Topics:

Evaluate Extract Options for a Downstream Deployment

To configure an Extract on the downstream mining database, consider the following guidelines:

- Multiple source databases can send their redo data to a single downstream database; however the downstream mining database can accept online redo logs from only one of those source databases. The rest of the source databases must ship archived logs.
- When online logs are shipped to the downstream database, real-time capture by Extract is possible. Changes are captured as though Extract is reading from the source logs. In order to accept online redo logs from a source database, the downstream mining database must have standby redo logs configured.
- When using a downstream mining configuration, the source database and mining database must be the same endian and same bit size, which is 64 bits. For example, if the source database was on Linux 64-bit, you can have the mining database run on Windows 64-bit, because they have the same endian and bit size.
- The initialization parameter `db_block_size` must be same between source database and mining database.

Prepare the Source Database for the Downstream Deployment

There must be an Extract user on the source database. Extract uses the credentials of this user to do metadata queries and to fetch column values as needed from the source database.

Add the credentials for connecting Extract to the source database from the Microservices Architecture web interface.

Topics:

Add Database Credentials to Connect to the Source Database

To create and run Extract and Replicat processes, you need to set up database credentials to connect Extract/Replicat users to the respective source or target databases.

1. Launch the **Administration Service** interface and log in.
2. Click **Configuration** from the **Application Navigation** pane.
3. Click the plus sign (+) sign next to Credentials.
4. Enter the following details in the displayed fields:

Database Credential Options	Description
Credential Domain	Specify a domain name to which the database credential is associated. For example, "OracleGoldenGate" is the default domain name, incase you don't specify a domain name.
Credential Alias	This is the alias for your database credential.
User ID	This is the username of the database user. For Oracle database, if you use the EZconnect syntax to connect to the database, then you can specify the value in this field in the following manner: <code>dbusername@hostname:port/service_name</code> <code>dbusername</code> is the database user name. <code>hostname</code> or IP address of the server where the database is running. <code>port</code> is the port number for connecting to the database server. Usually, this value is 1521. <code>service_name</code> is the name of the service provided in the tnsnames.ora file for the database connection.
Password	Password used by database user to log in to the database.

5. Click **Submit**.
6. Click the **Connect to database** icon to test that the connection is working correctly. If the connection is successful, the Connect to database icon turns blue.

When you successfully log into your database, you can add and manage checkpoint tables, transaction information (TRANSDATA), and heartbeat tables. All of the tables can be searched using the various search fields. As you type, the table is filtered and you can use the search button with the search text.

Configure Redo Transport from Source Database to Downstream Mining Database

To set up the transfer of redo log files from a source database to the downstream mining database, and to prepare the downstream mining database to accept these redo log files, perform the steps given in this topic.

The following summarizes the rules for supporting multiple sources sending redo to a single downstream mining database:

- Only one source database can be configured to send online redo to the standby redo logs at the downstream mining database. The `log_archive_dest_n` setting for this source database should not have a `TEMPLATE` clause.
- Source databases that are not sending online redo to the standby redo logs of the downstream mining database must have a `TEMPLATE` clause specified in the `log_archive_dest_n` parameter.
- Each of the source databases that sends redo to the downstream mining database must have a unique `DBID`. You can select the `DBID` column from the `v$database` view of these source databases to ensure that the `DBIDs` are unique.
- The `FAL_SERVER` value must be set to the downstream mining database. `FAL_SERVER` specifies the `FAL` (fetch archive log) server for a standby database. The value is a list of Oracle Net service names, which are assumed to be configured properly on the standby database system to point to the desired `FAL` servers. The list contains the net service name of any database that can potentially ship redo to the downstream database.
- When using redo transport, there could be a delay in processing redo due to network latency. For Extract, this latency is monitored by measuring the delay between LCRs received from source database and reporting it. If the latency exceeds a threshold, a warning message appears in the report file and a subsequent information message appears when the lag drops to normal values. The default value for the threshold is 10 seconds.

 **Note:**

The archived logs shipped from the source databases are called foreign archived logs. You must not use the recovery area at the downstream mining database to store foreign archived logs. Such a configuration is not supported by Extract. Foreign archived logs stored in the Flash Recovery Area (FRA) are not automatically deleted by RMAN jobs. These archived logs must be manually purged.

These instructions take into account the requirements to ship redo from multiple sources, if required. You must configure an Extract process for each of those sources.

To configure redo transport:

1. Configure database connection to connect the source database with the mining database.
2. Configure authentication at each source database and at the downstream mining database to support the transfer of redo data. Redo transport sessions are authenticated using either the Secure Sockets Layer (SSL) protocol or a remote login password file. If a source database has a remote login password file, copy it to the appropriate directory of the mining database system. The password file must be the same at all source databases, and at the mining database.
3. At each source database, configure one `LOG_ARCHIVE_DEST_n` initialization parameter to transmit redo data to the downstream mining database. Set the attributes of this parameter as shown in one of the following examples, depending on whether real-time or archived-log-only capture mode is to be used.

- Example for real-time capture at the downstream logmining server, where the source database sends its online redo logs to the downstream database:

```
ALTER SYSTEM
SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC NOREGISTER
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) DB_UNIQUE_NAME=dbmscap'
```

- Example for archived-log-only capture at the downstream logmining server:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC
NOREGISTER VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
TEMPLATE=/usr/oracle/log_for_dbms1/dbms1_arch_%t_%s_%r.log
DB_UNIQUE_NAME=dbmscap'
```

 **Note:**

When using an archived-log-only downstream mining database, you must specify a value for the `TEMPLATE` attribute. Oracle also recommends that you use the `TEMPLATE` clause in the source databases so that the log files from all remote source databases are kept separated from the local database log files, and from each other.

4. At the source database, set a value of `ENABLE` for the `LOG_ARCHIVE_DEST_STATE_n` initialization parameter that corresponds with the `LOG_ARCHIVE_DEST_n` parameter that corresponds to the destination for the downstream mining database, as shown in the following example:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

5. At the source database, and at the downstream mining database, set the `DG_CONFIG` attribute of the `LOG_ARCHIVE_CONFIG` initialization parameter to include the `DB_UNIQUE_NAME` of the source database and the downstream database, as shown in the following example:

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1,dbmscap) '
```

Prepare the Downstream Mining Database to Receive Online Redo Logs

A downstream mining database can accept both archived logs and online redo logs from a source database.

Topics:

Creating the Downstream Mining User Account

When using a downstream mining configuration, there must be an Extract mining user on the downstream database. The mining Extract process uses the credentials of this user to interact with the downstream logmining server.

The downstream mining user is specified by the `TRANLOGOPTIONS` parameter with the `MININGUSERALIAS` option.

See [Add Database Credentials to Connect to the Source Database](#) to assign the correct credentials for the version of your database.

Configure the Mining Database to Archive Local Redo Log Files

This procedure configures the downstream mining database to archive redo data in its online redo logs. These are redo logs that are generated at the downstream mining database.

Archiving must be enabled at the downstream mining database if you want to run Extract in real-time integrated capture mode, but it is also recommended for archive-log-only capture. Extract in integrated capture mode writes state information in the database. Archiving and regular backups will enable you to recover this state information in case there are disk failures or corruption at the downstream mining database.

To Archive Local Redo Log Files:

1. Alter the downstream mining database to be in archive log mode. You can do this by issuing the following

```
DDL.STARTUP MOUNT;
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
```

2. At the downstream mining database, set the first archive log destination in the `LOG_ARCHIVE_DEST_n` initialization parameter as shown in the following example:

```
ALTER SYSTEM SET
LOG_ARCHIVE_DEST_1='LOCATION=/home/arc_dest/local
VALID_FOR=(ONLINE_LOGFILE,PRIMARY_ROLE) '
```

Alternatively, you can use a command like this example:

```
ALTER SYSTEM SET  
LOG_ARCHIVE_DEST_1='LOCATION='USE_DB_RECOVERY_FILE_DEST'  
valid_for=(ONLINE_LOGFILE,PRIMARY_ROLE) '
```

 **Note:**

The online redo logs generated by the downstream mining database can be archived to a recovery area. However, you must not use the recovery area of the downstream mining database to stage foreign archived logs or to archive standby redo logs. For information about configuring a fast recovery area, see the Oracle Database Backup and Recovery User's Guide.

3. Enable the local archive destination.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

Configure the Wallet for the Downstream Mining Database

When TDE is enabled on source database and downstream database, then the source wallet or keys should be the same on the downstream mining database and the source database.

Follow these steps to copy the wallet directory from the source database to the downstream mining database:

1. Shutdown the downstream database using the shutdown immediate command.
2. Remove the wallet directory in the downstream database: `rm $OGG_DS_HOME/wallet/*`
3. Copy the `$OGG_DS_HOME/wallet/*` from the source database view to the downstream database view.
4. Restart the downstream database.
5. Run checksum on the source database view and downstream database view to ensure that it matches:

```
cksum $OGG_DS_HOME/wallet/*
```

Prepare a Downstream Mining Database for Real-time Capture

This procedure is only required if you want to use real-time capture at a downstream mining database. It is not required to use archived-log-only capture mode. To use real-time capture, it is assumed that the downstream database has already been configured to archive its local redo data as shown in [Configuring the Mining Database to Archive Local Redo Log Files](#).

Topics:

Create the Standby Redo Log Files

The following steps outline the procedure for adding standby redo log files to the downstream mining database. The following summarizes the rules for creating the standby redo logs:

- Each standby redo log file must be at least as large as the largest redo log file of the redo source database. For administrative ease, Oracle recommends that all redo log files at source database and the standby redo log files at the downstream mining database be of the same size.
- The standby redo log must have at least one more redo log group than the redo log at the source database, for each redo thread at the source database.

The specific steps and SQL statements that are required to add standby redo log files depend on your environment. See *Oracle Data Guard Concepts and Administration Guide* for detailed instructions about adding standby redo log files to a database.



Note:

If there are multiple source databases sending redo to a single downstream mining database, only one of those sources can send redo to the standby redo logs of the mining database. An Extract process that mines the redo from this source database can run in real-time mode. All other source databases must send only their archived logs to the downstream mining database, and the Extracts that read this data must be configured to run in archived-log-only mode.

To create the standby redo log files:

1. In SQL*Plus, connect to the source database as an administrative user.
2. Determine the size of the source log file. Make note of the results.

```
SELECT BYTES FROM V$LOG;
```

3. Determine the number of online log file groups that are configured on the source database. Make note of the results.

```
SELECT COUNT(GROUP#) FROM V$LOG;
```

4. Connect to the downstream mining database as an administrative user.
5. Add the standby log file groups to the mining database. The standby log file size must be at least the size of the source log file size. The number of standby log file groups must be at least one more than the number of source online log file groups. This applies to each instance (thread) in a RAC installation. So if you have "n" threads at the source database, each having "m" redo log groups, you should configure $n \times (m+1)$ redo log groups at the downstream mining database.

The following example shows three standby log groups.

```
ALTER DATABASE ADD STANDBY LOGFILE GROUP 3
('/oracle/dbs/slog3a.rdo', '/oracle/dbs/slog3b.rdo')
SIZE 500M; ALTER DATABASE ADD STANDBY LOGFILE
GROUP 4 ('/oracle/dbs/slog4.rdo', '/oracle/dbs/slog4b.rdo')
SIZE 500M; ALTER DATABASE ADD STANDBY LOGFILE GROUP 5 ('/oracle/dbs/slog5.rdo', '/
oracle/dbs/slog5b.rdo') SIZE 500M;
```

6. Confirm that the standby log file groups were added successfully.

```
SELECT GROUP#, THREAD#, SEQUENCE#, ARCHIVED, STATUS FROM V$STANDBY_LOG;
```

The output should be similar to the following:

```
GROUP# THREAD# SEQUENCE# ARC STATUS -----
3 0 0
YES UNASSIGNED 4 0 0 YES UNASSIGNED 5 0 0 YES UNASSIGNED
```

7. Ensure that log files from the source database are appearing in the location that is specified in the

```
LOCATION
```

attribute of the local

```
LOG_ARCHIVE_DEST_n
```

that you set. You might need to switch the log file at the source database to see files in the directory.

Configure the Database to Archive Standby Redo Log Files Locally

This procedure configures the downstream mining database to archive the standby redo logs that receive redo data from the online redo logs of the source database. Keep in mind that foreign archived logs should not be archived in the recovery area of the downstream mining database.

To Archive Standby Redo Logs Locally:

1. At the downstream mining database, set the second archive log destination in the `LOG_ARCHIVE_DEST_n` initialization parameter as shown in the following example.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='LOCATION=/home/arc_dest/srl_dbms1  
VALID_FOR=(STANDBY_LOGFILE,PRIMARY_ROLE) '
```

Oracle recommends that foreign archived logs (logs from remote source databases) be kept separate from local mining database log files, and from each other. You must not use the recovery area of the downstream mining database to stage foreign archived logs.

2. Enable the `LOG_ARCHIVE_DEST_2` parameter you set in the previous step as shown in the following example.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

Enable Downstream Extract to Work with ADG

In a cascaded downstream capture environment, the downstream database does not connect directly to the source database. It uses the Active Data Guard (ADG) as a reference.

Extract must be started using the sourceless option so that it does not connect to the source database and instead connects to ADG using `FETCHUSERID` or `FETCHUSERIDALIAS` when it needs to fetch any non-native datatypes. For example, `FETCH` operations are processed on the ADG database as this instance is open in read-only mode. Other operations that cannot be processed on the ADG instance, such as creating the dictionary build, are redirected from the ADG to the source database.

When registering a downstream Extract, Oracle GoldenGate connects to ADG as source database instead of the database where the redo originates. ADG redirection is supported for the following commands and parameters:

- `SCHEMATRANDATA`
- `TRANDATA`
- `FLUSH SEQUENCE`
- `TRACETABLE`
- `HEARTBEATTABLE`
- `REGISTER EXTRACT`

**Note:**

`SCHEMATRANDATA` and `TRANDATA`, even though the command is executed on the standby redo log, the actual log groups are created and maintained on the primary database where the actual DML operations take place.

ADG Redirection is available with Oracle Database 21c and higher. It also supports wildcard registration. The following example shows the Extract parameter file for the downstream Extract, when using ADG:

```
EXTRACT EXTDSO
NOUSERID
TRANLOGOPTIONS MININGUSERALIAS cgg_cdbDSC_src DOMAIN OracleGoldenGate
TRANLOGOPTIONS INTEGRATEDPARAMS (DOWNSTREAM_REAL_TIME_MINE Y)
FETCHUSERIDALIAS cgg_cdbADG_src DOMAIN OracleGoldenGate

EXTTRAIL cascade/ea
SOURCECATALOG CDBNORTH_PDB01
DDL INCLUDE MAPPED
TABLE HR.*;
```

Here are the steps to enable downstream Extract to work with ADG Standby:

1. Add an additional `LOG_ARCHIVE_DESTINATION_N` (LAD) on the ADG standby, as shown in the following example:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_3='service=mining_db_service_name ASYNC
NOREGISTER VALID_FOR(STANDBY_LOGFILES,STANDBY_ROLES)
DB_UNIQUE_NAME=3rd_db_unique_name' scope=both
```

This step transports and generates the `standby_logfiles` for an ADG standby.

2. Set the `LOG_ARCHIVE_CONFIG` on the ADG standby to ship the logs to the mining database, as shown in the following example:

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='dg_config' scope=both;
```

`db_config` is the database unique name of the first, second, and third databases.

3. On the mining database, set up the location to store the incoming `standby_logfiles` on the mining database:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='location= DB_RECOVERY_FILE_DEST
VALID_FOR=(STANDBY_LOGFILE,ALL_ROLES)' scope=both
```

The location is the database recovery file destination.

4. Run `LOG_ARCHIVE_CONFIG` on the mining database, so that the Extract process is able to read them on the mining database, as shown in the following example:

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='dg_config' scope=both
```

Here, `db_config` is the database unique name of the first, second, and third databases.

5. For a downstream Extract, you need to ensure that the database connections are appropriately configured. When registering the Extract, make sure that `DBLOGIN` connection is made to the ADG Standby, that is open for read-only activity.
6. To add the Extract and register it, use the following command:

```
DBLOGIN USERID ggadmin@cdbADG_src, PASSWORD ggadmin
MININGDBLOGIN USERID ggadmin@cgg_cdbDSC, password ggadmin
```

`cdbADG_src` is the ADG not primary.

`cgg_cdbDSC` is the mining database.

7. Now, register an Extract that uses the `NOUSERID` parameter:

```
ADD EXTRACT exte, INTEGRATED TRANLOG, BEGIN NOW REGISTER EXTRACT exte
DATABASE
```

8. After the Extract is registered, you can use this Extract to mine data and start the Extract normally.

Use Cases for Downstream Mining Configuration

Read about the different downstream mining configuration use cases.

Topics:

Case 1: Capture from One Source Database in Real-time Mode

This example captures changes from source database `DBMS1` by deploying an Extract at a downstream mining database `DBMSCAP`.

The example assumes that you created the necessary standby redo log files as shown in [Configure Extract for a Downstream Deployment](#).

This assumes that the following users exist:

- User `GGADM1` in `DBMS1` whose credentials Extract will use to fetch data and metadata from `DBMS1`. This user has the alias of `ggadm1` in the Oracle GoldenGate credential store and logs in as `ggadm1@dbms1`. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at the source database.
- User `GGADMCAP` in `DBMSCAP` whose credentials Extract will use to retrieve logical change records from the logmining server at the downstream mining database `DBMSCAP`. This user has the alias of `ggadmcap` in the Oracle GoldenGate credential store and logs in as `ggadmcap@dbmscap`. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at the mining database.

Topics:

Prepare the Mining Database to Archive its Local Redo

To prepare the mining database to archive its local redo:

1. The downstream mining database must be in archive log mode. You can do this by issuing the following DDL:

```
STARTUP MOUNT;  
ALTER DATABASE ARCHIVELOG;  
ALTER DATABASE OPEN;
```

2. At the downstream mining database, set `log_archive_dest_1` to archive local redo:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1='LOCATION=/home/arc_dest/local  
VALID_FOR=(ONLINE_LOGFILE, PRIMARY_ROLE) '
```

3. Enable `log_archive_dest_1`.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

Prepare the Mining Database to Archive Redo Received in Standby Redo Logs from the Source Database

To prepare the mining database to archive the redo received in standby redo logs from the source database:

1. At the downstream mining database, set `log_archive_dest_2` as shown in the following example:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='LOCATION=/home/arc_dest/srl_dbms1  
VALID_FOR=(STANDBY_LOGFILE, PRIMARY_ROLE) '
```

2. Enable `log_archive_dest_2` as shown in the following example:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

3. Set `DG_CONFIG` at the downstream mining database:

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1,dbmscap) '
```

Prepare the Source Database to Send Redo to the Mining Database

To prepare the source database to send redo to the mining database::

1. Make sure that the source database is running with the required compatibility:

```
select name, value from v$parameter where name = 'compatible';
```

The minimum compatibility setting required from integrated capture is 11.1.0.0.0.

2. Set `DG_CONFIG` at the source database:

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1,dbmscap) ';
```

3. Set up redo transport at the source database..

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC  
OPTIONAL NOREGISTER  
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

Set up Extract (ext1) on DBMSCAP

To set up Extract (ext1) on DBMSCAP:

- 1. Register Extract with the downstream mining database.** In the credential store, the alias name of `ggadm1` is linked to a user connect string of `ggadm1@dbms1`. The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

```
DBLOGIN USERIDALIAS ggadm1
```

```
MININGDBLOGIN USERIDALIAS ggadmcap
```

```
REGISTER EXTRACT ext1 DATABASE
```

- 2. Create Extract at the downstream mining database:**

```
ADD EXTRACT ext1 INTEGRATED TRANLOG BEGIN NOW
```

- 3. Edit Extract parameter file `ext1.prm`.** The following lines must be present to take advantage of real-time capture. In the credential store, the alias name of `ggadm1` is linked to a user connect string of `ggadm1@dbms1`. The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

```
USERIDALIAS ggadm1 TRANLOGOPTIONS MININGUSERALIAS ggadmcap TRANLOGOPTIONS  
INTEGRATEDPARAMS (downstream_real_time_mine Y)
```

- 4. Start Extract.**

```
START EXTRACT ext1
```

**Note:**

You can create multiple Extracts running in real-time Extract mode in the downstream mining database, as long as they all are capturing data from the same source database, such as capturing changes for database DBMS1 in the preceding example.

Case 2: Capture from Multiple Sources in Archive-log-only Mode

The following example captures changes from database `DBMS1` and `DBMS2` by deploying an Extract at a downstream mining database `DBMSCAP`.

It assumes the following users:

- User `GGADM1` in `DBMS1` whose credentials Extract will use to fetch data and metadata from `DBMS1`. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at `DBMS1`.
- User `GGADM2` in `DBMS2` whose credentials Extract will use to fetch data and metadata from `DBMS2`. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at `DBMS2`.
- User `GGADMCAP` in `DBMSCAP` whose credentials Extract will use to retrieve logical change records from the logmining server at the downstream mining database. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at the downstream mining database `DBMSCAP`.

This procedure also assumes that the downstream mining database is configured in archive log mode.

Topics:

Prepare the Mining Database to Archive its Local Redo

To prepare the mining database to archive its local redo:

1. The downstream mining database must be in archive log mode. You can do this by issuing the following DDL.

```
STARTUP MOUNT; ALTER DATABASE ARCHIVELOG; ALTER DATABASE OPEN;
```

2. At the downstream mining database, set `log_archive_dest_1` to archive local redo.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1='LOCATION=/home/arc_dest/local  
VALID_FOR=(ONLINE_LOGFILE, PRIMARY_ROLE)'
```

3. Enable `log_archive_dest_1`.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

4. Start Extract.

```
START EXTRACT ext1
```



Note:

You can create multiple Extracts running in real-time Extract mode in the downstream mining database, as long as they all are capturing data from the same source database, such as capturing changes for database `DBMS1` in the preceding example.

Prepare the Mining Database to Archive Redo from the Source Database

Set `DG_CONFIG` at the downstream mining database.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1,dbms2, dbmscap) '
```

Topics:

Prepare the First Source Database to Send Redo to the Mining Database

To prepare the first source database to send redo to the mining database:

1. Make certain that DBMS1 source database is running with the required compatibility.

```
select name, value from v$parameter where name = 'compatible'; NAME VALUE
-----
compatible 11.1.0.0.0
```

The minimum compatibility setting required from capture is 11.1.0.0.0.

2. Set `DG_CONFIG` at DBMS1 source database.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1, dbmscap)';
```

3. Set up redo transport at DBMS1 source database. The `TEMPLATE` clause is mandatory if you want to send redo data directly to foreign archived logs at the downstream mining database.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC
OPTIONAL NOREGISTER TEMPLATE='/usr/orcl/arc_dest/dbms1/
dbms1_arch_%t_%s_%r.log
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```



Note:

You can create multiple Extracts running in real-time Extract mode in the downstream mining database, as long as they all are capturing data from the same source database, such as capturing changes for database DBMS1 in the preceding example.

Prepare the Second Source Database to Send Redo to the Mining Database

To prepare the second source database to send redo to the mining database:

1. Make sure that DBMS2 source database is running with the required compatibility.

```
select name, value from v$parameter where name = 'compatible'; NAME VALUE
-----
compatible 11.1.0.0.0
```

The minimum compatibility setting required from capture is 11.1.0.0.0.

2. Set `DG_CONFIG` at DBMS2 source database.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms2, dbmscap)';
```

3. Set up redo transport at DBMS2 source database. The `TEMPLATE` clause is mandatory if you want to send redo data directly to foreign archived logs at the downstream mining database.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC
OPTIONAL NOREGISTER TEMPLATE='/usr/orcl/arc_dest/dbms2/
dbms2_arch_%t_%s_%r.log
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```



Note:

You can create multiple Extracts running in real-time Extract mode in the downstream mining database, as long as they all are capturing data from the same source database, such as capturing changes for database DBMS1 in the preceding example.

Set up Extracts at Downstream Mining Database

These steps set up Extract at the downstream database to capture from the archived logs sent by DBMS1 and DBMS2.

Case 3: Capturing from Multiple Sources with Mixed Real-time and Archive-log-only Mode

The following example captures changes from database DBMS1, DBMS2 and DBMS3 by deploying an Extract at a downstream mining database DBMSCAP.



Note:

This example assumes that you created the necessary standby redo log files as shown in [Prepare the Downstream Mining Database to Receive Online Redo Logs](#). It assumes the following users:

- User `GGADM1` in DBMS1 whose credentials Extract will use to fetch data and metadata from DBMS1. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at DBMS1.
- User `GGADM2` in DBMS2 whose credentials Extract will use to fetch data and metadata from DBMS2. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at DBMS2.

- User GGADM3 in DBMS3 whose credentials Extract will use to fetch data and metadata from DBMS3. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at DBMS3.
- User GGADMCAP in DBMSCAP whose credentials Extract will use to retrieve logical change records from the logmining server at the downstream mining database. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at the downstream mining database DBMSCAP.

This procedure also assumes that the downstream mining database is configured in archive log mode.

In this example, the redo sent by DBMS3 will be mined in real time mode, whereas the redo data sent from DBMS1 and DBMS2 will be mined in archive-log-only mode.

Topics:

Prepare the Mining Database to Archive its Local Redo

To prepare the mining database to archive its local redo:

1. The downstream mining database must be in archive log mode. You can do this by issuing the following DDL:

```
STARTUP MOUNT;
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
```

2. At the downstream mining database, set `log_archive_dest_1` to archive local redo.:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1='LOCATION=/home/arc_dest/local
VALID_FOR=(ONLINE_LOGFILE, PRIMARY_ROLE) '
```

3. Enable `log_archive_dest_1`.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

Prepare the Mining Database to Accept Redo from the Source Databases

Because redo data is being accepted in the standby redo logs of the downstream mining database, the appropriate number of correctly sized standby redo logs must exist. If you did not configure the standby logs, see [Create the Standby Redo Log Files](#).

1. At the downstream mining database, set the second archive log destination in the `LOG_ARCHIVE_DEST_n` initialization parameter as shown in the following example. This is needed to handle archive standby redo logs.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='LOCATION=/home/arc_dest/srl_dbms3
VALID_FOR=(STANDBY_LOGFILE, PRIMARY_ROLE) '
```

2. Enable the `LOG_ARCHIVE_DEST_STATE_2` initialization parameter that corresponds with the `LOG_ARCHIVE_DEST_2` parameter as shown in the following example:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

3. Set `DG_CONFIG` at the downstream mining database to accept redo data from all of the source databases.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1, dbms2, dbms3, dbmscap)'
```

Prepare the First Source Database to Send Redo to the Mining Database

To prepare the first source database to send redo to the mining database:

1. Make certain that DBMS1 source database is running with the required compatibility.

```
select name, value from v$parameter where name = 'compatible';
```

2. Set `DG_CONFIG` at DBMS1 source database:

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1, dbmscap)';
```

3. Set up redo transport at DBMS1 source database. The `TEMPLATE` clause is mandatory if you want to send redo data directly to foreign archived logs at the downstream mining database.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC  
OPTIONAL NOREGISTER TEMPLATE='/usr/orcl/arc_dest/dbms1/  
dbms1_arch_%t_%s_%r.log  
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

Prepare the Second Source Database to Send Redo to the Mining Database

To prepare the second source database to send redo to the mining database::

1. Make sure that DBMS2 source database is running with the required compatibility.

```
select name, value from v$parameter where name = 'compatible';
```

2. Set `DG_CONFIG` at DBMS2 source database:

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms2, dbmscap)';
```

3. Set up redo transport at DBMS2 source database. The `TEMPLATE` clause is mandatory if you want to send redo data directly to foreign archived logs at the downstream mining database.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC  
OPTIONAL NOREGISTER TEMPLATE='/usr/orcl/arc_dest/dbms2/  
dbms2_arch_%t_%s_%r.log  
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

Prepare the Second Source Database to Send Redo to the Mining Database

To prepare the second source database to send redo to the mining database::

1. Make sure that DBMS2 source database is running with the required compatibility.

```
select name, value from v$parameter where name = 'compatible';
```

2. Set DG_CONFIG at DBMS2 source database:

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms2, dbmscap)';
```

3. Set up redo transport at DBMS2 source database. The `TEMPLATE` clause is mandatory if you want to send redo data directly to foreign archived logs at the downstream mining database.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC  
OPTIONAL NOREGISTER TEMPLATE='/usr/orcl/arc_dest/dbms2/  
dbms2_arch_%t_%s_%r.log  
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

Prepare the Third Source Database to Send Redo to the Mining Database

To prepare the third source database to send redo to the mining database:

1. Make sure that DBMS3 source database is running with the required compatibility.

```
select name, value from v$parameter where name = 'compatible'; NAME VALUE  
-----  
compatible 11.1.0.0.0
```

The minimum compatibility setting required from capture is 11.1.0.0.0.

2. Set DG_CONFIG at DBMS3 source database:

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms3, dbmscap)';
```

3. Set up redo transport at DBMS3 source database. Because DBMS3 is the source that will send its online redo logs to the standby redo logs at the downstream mining database, do not specify a `TEMPLATE` clause.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC  
OPTIONAL NOREGISTER  
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

Topics:

Set up Extracts at Downstream Mining Database

These steps set up Extract at the downstream database to capture from the archived logs sent by DBMS1 and DBMS2.

Topics:

Set up Extract (ext1) to Capture Changes from Archived Logs Sent by DBMS1

Perform the following steps on the DBMSCAP downstream mining database:

1. Register Extract with DBMSCAP for the DBMS1 source database. In the credential store, the alias name of ggadm1 is linked to a user connect string of ggadm1@dbms1. The alias name of ggadmcap is linked to a user connect string of ggadmcap@dbmscap.

```
DBLOGIN USERIDALIAS ggadm1
MININGDBLOGIN
USERIDALIAS ggadmcap
REGISTER EXTRACT ext1 DATABASE
```

2. Add Extract at the mining database DBMSCAP:

```
ADD EXTRACT ext1 INTEGRATED TRANLOG BEGIN NOW
```

3. Edit the Extract parameter file ext1.prm. In the credential store, the alias name of ggadm1 is linked to a user connect string of ggadm1@dbms1. The alias name of ggadmcap is linked to a user connect string of ggadmcap@dbmscap:

```
USERIDALIAS ggadm1 TRANLOGOPTIONS MININGUSERALIAS
ggadmcap TRANLOGOPTIONS INTEGRATEDPARAMS (downstream_real_time_mine N)
```

4. Start Extract:

```
START EXTRACT ext1
```

Set up Extract (ext2) to Capture Changes from Archived Logs Sent by DBMS2

Perform the following steps on the DBMSCAP downstream mining database:

1. Register Extract with the mining database for source database DBMS2. In the credential store, the alias name of ggadm2 is linked to a user connect string of ggadm2@dbms2. The alias name of ggadmcap is linked to a user connect string of ggadmcap@dbmscap.

```
DBLOGIN USERIDALIAS ggadm2
MININGDBLOGIN USERIDALIAS ggadmcap
REGISTER EXTRACT ext2 DATABASE
```

2. Create Extract at the mining database:

```
ADD EXTRACT ext2 INTEGRATED TRANLOG, BEGIN NOW
```

3. Edit the Extract parameter file `ext2.prm`. In the credential store, the alias name of `ggadm2` is linked to a user connect string of `ggadm2@dbms2`. The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`:

```
USERIDALIAS ggadm2 TRANLOGOPTIONS MININGUSERALIAS ggadmcap TRANLOGOPTIONS
INTEGRATEDPARAMS (downstream_real_time_mine N)
```

4. Start Extract:

```
START EXTRACT ext2
```

Set up Extract (ext3) to Capture Changes in Real-time Mode from Online Logs Sent by DBMS3
Perform the following steps on the DBMSCAP downstream mining database:

1. Register Extract with the mining database for source database DBMS3. In the credential store, the alias name of `ggadm3` is linked to a user connect string of `ggadm3@dbms3`. The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

```
DBLOGIN USERID ggadm3
MININGDBLOGIN USERID ggadmcap
REGISTER EXTRACT ext3 DATABASE
```

2. Create Extract at the mining database:

```
ADD EXTRACT ext3 INTEGRATED TRANLOG, BEGIN NOW
```

3. Edit the Extract parameter file `ext3.prm`. In the credential store, the alias name of `ggadm3` is linked to a user connect string of `ggadm3@dbms3`. The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`:

```
USERIDALIAS ggadm3
TRANLOGOPTIONS MININGUSERALIAS ggadmcap TRANLOGOPTIONS
INTEGRATEDPARAMS (downstream_real_time_mine N)
```

4. Start Extract:

```
START EXTRACT ext3
```



Note:

You can create multiple Extracts running in real-time integrated capture mode in the downstream mining database, as long as they all are capturing data from the same source database, such as all capturing for database DBMS3 in the preceding example.

Positioning Extract to a Specific Start Point

You can position the Extract to a specific start point in the transaction logs using the `ADD/ALTER EXTRACT` commands:

```
{ADD | ALTER EXTRACT} group, LOGNUM log_num, LOGPOS log_pos
```

- *group* is the name of the Oracle GoldenGate Extract group for which the start position is required.
- *LOGNUM* is the log file number. For example, if the required log file name is `test.000034`, the *LOGNUM* value is 34. Extract will search for this log file. The `ADD EXTRACT` command will fail if the *LOGNUM* value contains zeroes preceding the value. For example, `ADD EXTRACT ext1, TRANLOG, LOGNUM 000001, LOGPOS 0` will fail. Instead, set *LOGNUM* to 1 for this example to succeed.
- *LOGPOS* is an event offset value within the log file that identifies a specific transaction record. Event offset values are stored in the header section of a log record. To position at the beginning of a `binlog` file, set the *LOGPOS* as 0.

In MySQL logs, an event offset value can be unique only within a given binary file. The combination of the position value and a log number will uniquely identify a transaction record. Maximum Log number length is 8 bytes unsigned integer and Maximum Log offset length is 8 bytes unsigned integer. Log number and Log offset are separated by a pipe ('|') delimiter. Transactional records available after this position within the specified log will be captured by Extract. In addition, you can position an Extract using a timestamp.

6

Distribute

Learn about the Distribution Service, how to add a distribution path, how to add a target-initiated distribution paths, and managing distribution paths.

About Distribution Service and Distribution Path

The Distribution Service is accessible from the Service Manager home page or you can directly specify the URL in a web browser.

Log in to the Distribution Service for the associated deployment. From the Distribution Service home page you can see a dashboard that displays the path connecting the Extract and Replicat processes. You can add a distribution path or data streams from this interface.

Use the dashboard to perform the following operations.

Action	Reference
<ul style="list-style-type: none">• Add distribution paths• Add data streams• View path details• Start or Stop the path• Reposition the path• Enable sharding using filters• Set or customize the DML filtering• Set the DDL filtering• Set or customize Procedure filtering• Customize Tag filtering• Delete a Path	<p>See:</p> <ul style="list-style-type: none">Add a Distribution PathUsing the Path ActionsManage Distribution Paths <p>Also see:</p> <p><code>ALTER DISTPATH</code> command options.</p>

About Distribution Paths

A path is used to send trail data between two data endpoints of a deployment. You can add, monitor, reposition, and manage these paths using the Distribution Service. This topic discusses the steps to create a distribution path (`DISTPATHS`).

A distribution path defines the route for the trail to send and receive data for different topologies. Oracle GoldenGate uses the **target authentication method** to define the method for connecting source and target deployments. The options for setting up the target authentication method are as follows:

- **USER ID ALIAS target authentication:** On the target deployment, a user with Operator role is created and then the credentials of this user are added as credentials in the source Oracle GoldenGate deployment. When using the `USERIDALIAS` method, while creating the Distribution Path, the value of target authentication method is set to **Password**. The **WSS** (secure web socket) protocol is used for this type of distribution path.
- **Certificate target authentication:** In this case, the distribution path uses trusted CA certificates to access the target deployment. The target authentication method that is set up while creating the Distribution Path is **Certificate**. The **WSS** (secure web socket) protocol is used for this type of distribution path.

- **OAuth target authentication:** In this case, the Oracle GoldenGate user authentication is outsourced to an OAuth service such as IDCS and IAM as cloud-based identity providers and OAM as an on-premise identity provider.

Distribution Path Streaming Protocols

You will need to configure a protocol for the Distribution Path to transfer trail files over the network. This configuration is done when you create a Distribution Path in the Distribution Service.

For details about selecting the streaming protocol, see [Add a Distribution Path](#).

While setting up the Distribution Path, if you select `USERIDALIAS` as the **target authentication method**, then you can select from one of the following protocols that would be used for streaming trail data over the network:

- Secure Web Socket (`wss`): Secure and recommended protocol.
- Web Sockets (`ws`): Unsecure deployments.
- Oracle GoldenGate protocol (`ogg`): Provides interoperability with a non-microservices deployment.

The following matrix provides the combinations of streaming protocols used with Oracle GoldenGate Microservices:

Source/Target	MA Non-Secure	MA Non-Secure with NGINX	MA Secure	Classic Architecture
MA Non-secure	Distribution path with ws protocol	Distribution path with wss protocol	Distribution path with wss protocol	Distribution Path with ogg protocol
MA Non-secure with NGINX	Distribution path with ws protocol	Distribution path with wss protocol	Distribution path with wss protocol	Distribution Path with ogg protocol
MA Secure	Distribution path with ws protocol	Distribution path with wss protocol	Distribution path with wss protocol	Distribution Path with ogg protocol
Classic Architecture	Oracle GoldenGate pump Extract, connect to the Receiver Service port	Need expose target Receiver Service port, then use Oracle GoldenGate pump Extract to connect to the Receiver port. directly	NA	Regular Oracle GoldenGate pump Extract

Also see, [Configure Reverse Proxy with NGINX to Access OGGMA](#).

Add a Distribution Path

A path is created to send the transaction of data from the Extract to the Replicat. You can create a new path from the Distribution Service.

To add a path to set the trail for the source deployment:

1. Log in to the **Distribution Service**.
2. Click the plus (+) sign next to Path on the Distribution Service home page.
The Add Path page is displayed.
3. Enter the details as follows:

Options	Description
Path Name	Select a name for the path.
Description	Provide a description. For example, the name of the Extract and Replicat names.
Reverse proxy enabled?	Select to use reverse proxy. To know more about configuring you reverser proxy servers, see Reverse Proxy Support in <i>Oracle GoldenGate Security Guide</i>
Use Basic Authentication	Select to add a credential to the target URI creating basic MA authentication.
Use Digest Authorization	Select this option to set the Distribution Service to use digest authorization to communicate with the Receiver Service.

**Note:**

Both the Distribution Service and Receiver Service must have Digest Authorization for the path, otherwise the path is killed.

Source: <i>Trail Name</i>	Select the Extract name from the drop-down list, which populates the trail name automatically. If it doesn't, enter the trail name that you provided while adding the Extract.
Generated Source URI:	A URI is automatically generated for the trail based on the Extract information you provided. You can edit this URI by clicking the pencil, then modifying the source. Typically, you will need to edit the URI if you want to use reverse proxy.
Target Authentication Method	<p>Select the authentication method for the target URI.</p> <p>Authentication options are OAuth, Certificate, UserID Alias.</p> <p>Use the OAuth if the source and target deployments are IDCS-enabled. This option uses the client credentials for authentication from the Distribution Service to the Receiver Service.</p>

Options	Description
Target	<p>Enter the target endpoint of the path.</p> <p>From the drop-down list, select your data transfer protocol. The default option is wss (secure web socket). Specify the following details when you select this option:</p> <ul style="list-style-type: none">• Target Host: Enter the URL of the target host, for example, localhost, if the target is on the same system.• Port Number: You may enter the port number of the Receiver Service and the trail name of the Replicat you created earlier. However, it's not mandatory. The port is the Manager port number for Classic Architecture.• Trail Name: Path takes the source trail and sends the data to a target trail given here, which can be consumed by any Replicats created later.• Domain: Name of the target domain.• Alias: User alias of the target domain. <p>You can also choose ogg or ws (web socket) protocol.</p> <p>For the ogg protocol, you need to specify only the target host, port number, and trail file name.</p> <p>For the ws protocol, the options are the same as the wss protocol.</p>
Generated Target URI	<p>A target URI is automatically generated for the trail based on the target authentication method and target you provided. You can edit this URI by clicking the pencil, then modifying the target.</p>
Target Encryption Algorithm	<p>Select the encryption algorithm for the target trail. Options include NONE, AES128, AES192, AES256.</p>
Target Encryption Keyname	<p>Specify a logical name for the encryption key based on the specified type of target encryption algorithm.</p>
Enable Network Compression	<p>Set the compression threshold value if you enable this option.</p>
Compression Threshold	<p>Option appears when you enable the network compression. Specify the compression threshold value.</p>
Sequence Length	<p>The length of the trail sequence number.</p>
Trail Size (MB)	<p>The maximum size of a file in a trail.</p>
Encryption Profile	<p>Name of the encryption profile associated with the path.</p>
Configure Trail Format	<p>Toggle this switch to enable and configure the trail file format.</p>
Type	<p>Select one of these types of trail file formats:</p> <ul style="list-style-type: none">• Plain Text• XML• SQL

Options	Description
Compatible With	Select the utility that is compatible with the trail file. Options are: <ul style="list-style-type: none"> • BCP • SQLLOADER • COMCAST
Timestamp Precision	Specify the timestamp precision value for the trail file.
Extra Columns	Includes placeholders for additional columns at the end of each record. Use this option when a target table has more columns than the source table. Specify a value between 1 and 9.
Include SYSKEY	Select this option incase your Replicat configuration includes tables with <code>SYSKEY</code> .
Quote Style	Select the quote style depending on the database requirements.
Include Column Name?	Enable this option to include column names in the trail file.
Null Is Space?	Select this option to indicate that any null values in the trail file is a space.
Include Place Holder?	Outputs a placeholder for missing columns.
Include Header Fields?	Select to include header fields in the trail file.
Delimiter	An alternative delimiter character.
Use Qualified Name?	Select to use the fully qualified name of the parameter file.
Include Transaction Info?	Enable to to include transaction information.
Encryption Profile	Section
Begin	Select the point from where you need to log data. You can select the following options from the drop-down list: <ul style="list-style-type: none"> • Now • Custom Time • Position is Log (default)
Source Sequence Number	Select the sequence number of the trail from source deployment Extract.
Source RBA Offset	This setting provides the Relative Byte Address (RBA) offset value which is the point in the trail file (in bytes) from where you want the process to start.
Critical	The default value is false. If set to true, this indicates that the distribution path is critical to the deployment.
Auto Restart	The default value is false. If set to true, the distribution path restarts automatically if it's terminated.
Auto Restart Options	Section
Retries	The number of times to try an restart the task (path process).
Delay	The duration interval to wait between retries.

Rule Configuration	Description
Enable filtering	<p>If you enable filtering by selecting it from the toggle button and click the Add Rule button, you'll see the Rule Definition dialog box.</p> <ul style="list-style-type: none"> Rule Name Rule Action: Select either Exclude or Include Filter Type: Select from the following list of options: <ul style="list-style-type: none"> Object Type: Select from three object types: DML, DDL, and Procedure Object Names: Select this option to provide an existing object name. A 3-part naming convention depends on whether you are using CDB. With CDB, you need to use a 3-part naming convention, otherwise a 2-part convention is mandatory. 3-part convention includes container, <i>schema</i>, <i>object</i>. 2-part convention includes <i>schema</i>, <i>object name</i>. Procedure Feature Name: Select this option to filter, based on existing procedure feature name. Column Based: If you select this option, you are presented with the option to enter the table and column name to which the rule applies. You can filter out using column value with LT, GT, EQ, LE, GE, NE conditions. You can also specify if you want to have before image or after image in filtered data. Tag: Select this option to set the filter based on tags. Chunk ID: Displays the configuration details of database shards, however, the details can't be edited. Negate: Select this check box if you need to negate any existing rule. <p>You can also see the JSON script for the rule by clicking the JSON tab.</p>
Additional Options	Description
Eof Delay (cent sec)	You can specify the Eof Delay in centiseconds. On Linux platforms, the default settings can be retained. However, on non-Linux platforms, you may need to adjust this setting for high bandwidth, high latency networks, or for networks that have Quality of Service (QoS) settings (DSCP and Time of Service (ToS)).
Checkpoint Frequency	Frequency of the path that is taking the checkpoint (in seconds).
TCP Flush Bytes	Enter the TCP flush size in bytes.
TCP Flush Seconds	Enter the TCP flush interval in seconds.

Additional Options	Description
TCP Options	Section
DSCP	Select the Differentiated Services Code Point (DSCP) value from the drop-down list, or search for it from the list.
TOS	Select the Type of service (TOS) value from the drop-down list.
TCP_NODELAY	Enable this option to prevent delay when using the Nagle's option.
Quick ACK	Enable this option to send quick acknowledgment after receiving data.
TCP_CORK	Enable this option to allow using the Nagle's algorithm cork option.
System Send Buffer Size	You can set the value for the send buffer size for flow control.
System Receive Buffer Size	You can set the value for the receive buffer size for flow control.
Keep Alive	Timeout for keep-alive.

4. Click **Create Path** or **Create and Run**, as required. Select **Cancel** if you need to get out of the Add Path page without adding a path.

Once the path is created, you'll be able to see the new path in the Overview page of the Distribution Service.


Add a Target-Initiated Distribution Path

To know more about target-initiated distribution paths, see Using Target-Initiated Distribution Paths in MA.

To create a target-initiated distribution path, perform the following steps:

1. Log in to the Receiver Server.
2. Click the + sign on the home page to start adding a path.
3. The following table lists the options to set up the path:

Options	Description
Path Name	Select a name for the path.
Description	Provide a description. For example, the name of the Extract and Replicat names.
Reverse proxy enabled?	Select to use reverse proxy. To know more about configuring your reverse proxy servers, see Reverse Proxy Support in <i>Oracle GoldenGate Security Guide</i> .
Use Basic Authentication	Select to add a credential to the target URI creating basic MA authentication.

Options	Description
Use Digest Authorization	Select this option to set the Distribution Service to use digest authorization to communicate with the Receiver Service.
	<div>  Note: Both the Distribution Service and Receiver Service must have Digest Authorization for the path, otherwise the path is killed. </div>
Source: <i>Trail Name</i>	Select the Extract name from the drop-down list, which populates the trail name automatically. If it doesn't, enter the trail name that you provided while adding the Extract.
Generated Source URI:	A URI is automatically generated for the trail based on the Extract information you provided. You can edit this URI by clicking the pencil, then modifying the source. Typically, you will need to edit the URI if you want to use reverse proxy.
Target Authentication Method	<p>Select the authentication method for the target URI. Authentication options are OAuth, Certificate, UserID Alias.</p> <p>Use the OAuth if the source and target deployments are IDCS-enabled. This option uses the client credentials for authentication from the Distribution Service to the Receiver Service.</p>

Options	Description
Target	<p>Enter the target endpoint of the path.</p> <p>From the drop-down list, select your data transfer protocol. The default option is wss (secure web socket). Specify the following details when you select this option:</p> <ul style="list-style-type: none"> • Target Host: Enter the URL of the target host, for example, localhost, if the target is on the same system. • Port Number: You may enter the port number of the Receiver Service and the trail name of the Replicat you created earlier. However, it's not mandatory. The port is the Manager port number for Classic Architecture. • Trail Name: Path takes the source trail and sends the data to a target trail given here, which can be consumed by any Replicats created later. • Domain: Name of the target domain. • Alias: User alias of the target domain. <p>You can also choose ogg or ws (web socket) protocol.</p> <p>For the ogg protocol, you need to specify only the target host, port number, and trail file name.</p> <p>For the ws protocol, the options are the same as the wss protocol.</p>
Generated Target URI	A target URI is automatically generated for the trail based on the target authentication method and target you provided. You can edit this URI by clicking the pencil, then modifying the target.
Target Encryption Algorithm	Select the encryption algorithm for the target trail. Options include NONE, AES128, AES192, AES256.
Target Encryption Keyname	Specify a logical name for the encryption key based on the specified type of target encryption algorithm.
Enable Network Compression	Set the compression threshold value if you enable this option.
Compression Threshold	Option appears when you enable the network compression. Specify the compression threshold value.
Sequence Length	The length of the trail sequence number.
Trail Size (MB)	The maximum size of a file in a trail.
Encryption Profile	Name of the encryption profile associated with the path.
Configure Trail Format	Toggle this switch to enable and configure the trail file format.
Type	<p>Select one of these types of trail file formats:</p> <ul style="list-style-type: none"> • Plain Text • XML • SQL

Options	Description
Compatible With	Select the utility that is compatible with the trail file. Options are: <ul style="list-style-type: none"> • BCP • SQLLOADER • COMCAST
Timestamp Precision	Specify the timestamp precision value for the trail file.
Extra Columns	Includes placeholders for additional columns at the end of each record. Use this option when a target table has more columns than the source table. Specify a value between 1 and 9.
Include SYSKEY	Select this option incase your Replicat configuration includes tables with <code>SYSKEY</code> .
Quote Style	Select the quote style depending on the database requirements.
Include Column Name?	Enable this option to include column names in the trail file.
Null Is Space?	Select this option to indicate that any null values in the trail file is a space.
Include Place Holder?	Outputs a placeholder for missing columns.
Include Header Fields?	Select to include header fields in the trail file.
Delimiter	An alternative delimiter character.
Use Qualified Name?	Select to use the fully qualified name of the parameter file.
Include Transaction Info?	Enable to to include transaction information.
Encryption Profile	Section
Begin	Select the point from where you need to log data. You can select the following options from the drop-down list: <ul style="list-style-type: none"> • Now • Custom Time • Position is Log (default)
Source Sequence Number	Select the sequence number of the trail from source deployment Extract.
Source RBA Offset	This setting provides the Relative Byte Address (RBA) offset value which is the point in the trail file (in bytes) from where you want the process to start.
Critical	The default value is false. If set to true, this indicates that the distribution path is critical to the deployment.
Auto Restart	The default value is false. If set to true, the distribution path restarts automatically if it's terminated.
Auto Restart Options	Section
Retries	The number of times to try an restart the task (path process).
Delay	The duration interval to wait between retries.

Rule Configuration	Description
Enable filtering	<p>If you enable filtering by selecting it from the toggle button and click the Add Rule button, you'll see the Rule Definition dialog box.</p> <ul style="list-style-type: none">• Rule Name• Rule Action: Select either Exclude or Include• Filter Type: Select from the following list of options:<ul style="list-style-type: none">– Object Type: Select from three object types: DML, DDL, and Procedure– Object Names: Select this option to provide an existing object name. A 3-part naming convention depends on whether you are using CDB. With CDB, you need to use a 3-part naming convention, otherwise a 2-part convention is mandatory. 3-part convention includes container, <i>schema</i>, <i>object</i>. 2-part convention includes <i>schema</i>, <i>object name</i>.– Procedure Feature Name: Select this option to filter, based on existing procedure feature name.– Column Based: If you select this option, you are presented with the option to enter the table and column name to which the rule applies. You can filter out using column value with LT, GT, EQ, LE, GE, NE conditions. You can also specify if you want to have before image or after image in filtered data.– Tag: Select this option to set the filter based on tags.– Chunk ID: Displays the configuration details of database shards, however, the details can't be edited.• Negate: Select this check box if you need to negate any existing rule. <p>You can also see the JSON script for the rule by clicking the JSON tab.</p>
Additional Options	Description
Eof Delay (cent sec)	You can specify the Eof Delay in centiseconds. On Linux platforms, the default settings can be retained. However, on non-Linux platforms, you may need to adjust this setting for high bandwidth, high latency networks, or for networks that have Quality of Service (QoS) settings (DSCP and Time of Service (ToS)).
Checkpoint Frequency	Frequency of the path that is taking the checkpoint (in seconds).
TCP Flush Bytes	Enter the TCP flush size in bytes.

Additional Options	Description
TCP Flush Seconds	Enter the TCP flush interval in seconds.
TCP Options	Section
DSCP	Select the Differentiated Services Code Point (DSCP) value from the drop-down list, or search for it from the list.
TOS	Select the Type of service (TOS) value from the drop-down list.
TCP_NODELAY	Enable this option to prevent delay when using the Nagle's option.
Quick ACK	Enable this option to send quick acknowledgment after receiving data.
TCP_CORK	Enable this option to allow using the Nagle's algorithm cork option.
System Send Buffer Size	You can set the value for the send buffer size for flow control.
System Receive Buffer Size	You can set the value for the receive buffer size for flow control.
Keep Alive	Timeout for keep-alive.

**Note:**

The the protocol options in Use Basic Authentication are `wss` and `ws` only for target-initiated distribution paths, unlike regular distribution paths, which provide `ogg` and `udt` options.

For target-initiated distribution paths, the use case for the `ws` and `wss` protocols is explained in the following table:

X	Target Deployment (Non-Secure)	Target Deployment (Secure)
Source Deployment (Non-secure)	<code>ws</code>	<code>ws</code>
Source Deployment (Secure)	<code>wss</code>	<code>wss</code>

The `wss` protocol must be specified whenever the source deployment (Distribution Server host) has been configured with security enabled. The secured communication channel can be created using an SSL certificate in a client Wallet, even if the target deployment (Receiver Server host) has disabled security.

Limitations

Here are the limitations when working with target-initiated paths:

- There is no support for interaction between legacy and secure deployments using this mode of operation.
- No support for `ogg` nor `udt` protocols. Only `ws` and `wss` protocols are supported.
- It is possible to only get information and stop a target-initiated distribution path on Distribution Server and after the path stops, it is not be visible on the Distribution Server.

You can also set up target-initiated distribution paths using the Admin Client. For command options, see the Admin Client commands `ADD RECVPATH`, `ALTER RECVPATH`, `INFO RECVPATH`, `DELETE RECVPATH`, `START RECVPATH` in Admin Client Command Line Interface Commands.

Manage Distribution Paths

Learn about managing distribution paths.

Manage Distribution Paths

Once a new path is added, you can perform actions such as stop or pause a path, view reports and statistics, reposition the path, change its filtering, and delete a path, if required.

On the Overview page of the Distribution Server, click the **Action** button adjacent to the path. From the drop-down list, use the following path actions:

- **Details:** Use this option to view details of the path. You can view the path information including the source and target. You can also edit the description of the path. Statistical data is also displayed including LCR Read from Trails, LCR Sent, LCR Filtered, DDL, Procedure, DML inserts, updates, and deletes, and so on. You can also update the App Options and TCP Options.
- **Stop:** Use this option to stop a path. If the path isn't started, the Start option is displayed rather than the Stop option. You can stop a target-initiated distribution path only from the Distribution Server. Once you stop the path, it'll not be available on the Distribution Server.
- **Stop (in the background):** This option stops the path in the background, without engaging the interface. For this option also, the Start (in background) option is displayed in case the path isn't started.
- **Delete:** Use this option to delete a path. Click Yes on the confirmation screen to complete path deletion.
- **Reposition:** Use this option to change the Source Sequence Number and Source RBA Offset.
- **Change Filtering:** Use this option to enter sharding, DML filtering, DDL filtering, Procedure filtering, and Tag filtering options.

Depending on the action you select, you can see the change in status at the bottom of the Overview page.

Reposition a Path

You can reposition a path as required. To reposition a distribution path or target-initiated distribution path:

1. From the Distribution Service home page, click **Distribution Path** to open the Distribution Paths page.
2. Click the **Action** button for the path and select **Reposition** from the drop-down list. The Reposition dialog box is displayed.
3. Specify the source trail Sequence Number and the Source RBA Offset.
4. Click **Apply**.

Change the Path Filtering

If you want to change the filter settings for an existing path, the steps are mostly the same as those for creating the filtering for a new path.

From the left navigation pane of the Distribution Service home page, click **Distribution Path**.

For the specific distribution path, click **Action**. From the drop-down list, click **Change Filtering**.

Rule Configuration	Task
Add paths	<p>If you enable filtering by selecting it from the toggle button and click Add Rule, you'll see the Rule Definition dialog box.</p> <ul style="list-style-type: none">• Rule Name• Rule Action: Select either Exclude or Include• Filter Type: Select from the following list of options:<ul style="list-style-type: none">– Object Type: Select from three object types: DML, DDL, and Procedure– Object Names: Select this option to provide an existing object name. A 3-part naming convention depends on whether you are using CDB. With CDB, you need to use a 3-part naming convention, otherwise a 2-part convention is mandatory. 3-part convention includes container, schema, object. 2-part convention includes schema, object name.– Procedure Feature Name: Select this option to filter, based on existing procedure feature name.– Column Based: If you select this option, you are presented with the option to enter the table and column name to which the rule applies. You can filter out using column value with LT, GT, EQ, LE, GE, NE conditions. You can also specify if you want to have before image or after image in filtered data.– Tag: Select this option to set the filter based on tags.– Chunk ID: Displays the configuration details of database shards, however, the details can't be edited.• Negate: Select this check box if you need to negate any existing rule. <p>You can also see the JSON script for the rule by clicking the JSON tab.</p>

After you add a rule, it is listed in **Inclusion Rules**. You can delete rules or edit them. When you edit a rule, you have the same options as adding a rule with the following added filters:

Options	Description
OR AND	Select one logical operator.
Chunk ID	Edit or delete the database shard settings if sharding is used.
Object Type:	Edit or delete the type of object for the rule.

If you want to change the filter settings for an existing path, the steps are mostly the same as those for creating the filtering for a new path.

On the Distribution Service home page, click **Action** for the path. From the drop-down list, click Change Filtering.

Rule Configuration	Task
Add paths	<p>If you enable filtering by selecting it from the toggle button and click Add Rule, you'll see the Rule Definition dialog box.</p> <ul style="list-style-type: none">• Rule Name• Rule Action: Select either Exclude or Include• Filter Type: Select from the following list of options:<ul style="list-style-type: none">– Object Type: Select from three object types: DML, DDL, and Procedure– Object Names: Select this option to provide an existing object name. A 3-part naming convention depends on whether you are using CDB. With CDB, you need to use a 3-part naming convention, otherwise a 2-part convention is mandatory. 3-part convention includes container, schema, object. 2-part convention includes schema, object name. <div> Note: Starting with Oracle GoldenGate 23ai, CDBs are only used with Downstream Extracts.</div> <ul style="list-style-type: none">– Procedure Feature Name: Select this option to filter, based on existing procedure feature name.– Column Based: If you select this option, you are presented with the option to enter the table and column name to which the rule applies. You can filter out using column value with LT, GT, EQ, LE, GE, NE conditions. You can also specify if you want to have before image or after image in filtered data.– Tag: Select this option to set the filter based on tags.– Chunk ID: Displays the configuration details of database shards, however, the details can't be edited.• Negate: Select this check box if you need to negate any existing rule. <p>You can also see the JSON script for the rule by clicking the JSON tab.</p>

After you add a rule, it is listed in Inclusion Rules. You can delete rules or edit them. When you edit a rule, you have the same options as adding a rule with the following added filters:

Options	Description
OR AND	Select one logical operator.
Chunk ID	Edit or delete the database shard settings if sharding is used.
Object Type:	Edit or delete the type of object for the rule.

For setting up the filtering options in a Distribution path or the Receiver path, see the `ALTER DISTPATH` and `ALTER RECVPATH` in the *Command Line Interface Reference for Oracle GoldenGate*.

Review the Distribution Path Information

You can constantly monitor the activity of the path on the Distribution path information page. To access the Distribution Path information page, click Distribution Paths, select the distribution path name that you need to view. The Basic Information of the distribution path is displayed. You can also click the Path Information option under the distribution path name in the left navigation pane.

This page displays all details of the associated path and allows you to edit or modify various options. The editable options have a pencil icon available with it. The information displayed includes:

- From the basic information, you can change the Target Encryption Algorithm, , Trail Size, configure trail format, enable or disable the Critical option depending on whether the path is considered critical to the deployment, change Auto Restart value.
- From the Encryption section, you can edit the encryption profile name, and apply a new master key if required.
- The Advanced Options including Enable Network Compression, EOF delay, flush, and TCP that you configured. You can change any or all of these options, then apply to the path.

7

Replicat

Learn about the Replicat process, its types, and steps to add a replicat, and other tasks associated with Replicat.

Quick Tour of the Administration Service Overview Page

When you click the Administrator Server link on the Service Manager home page, the login page for the Administration Server is displayed. After logging in, you can configure Extract and Replicat processes from this Web UI.

The Administration Server home page is used to add Extracts and Replicats. The table on the home page displays the severity of critical events. You can also use the left-navigation pane to access various configuration details, a list of severity issues with their diagnosis, and a list of administrators.

Now, that you have an overview of the Administration Server home page, let's understand some of the key actions that you can perform from this page.

Action	Description
View the home page in tabular format	Use the Table Layout swivel to turn the tabular format on and off.
View Extracts and Replicats	The statistical representation the home page displays current state of Extracts and Replicats (Starting, Running, Stopped, Abended, Killed)
Add an Extract	See How to Add an Extract for a Deployment
Create a Replicat	See How to Add a Replicat
Stop and start Extracts	Using Extract Actions
Stop and start Replicats	See Using Replicat Actions
View and search critical events	Monitor severity of events using the Critical Events table and also search for specific events, if required.

About Replicat

Replicat is a process that delivers data to a target system. It reads the trail file on the target database, reconstructs the DML or DDL operations, and applies them to the target database.

The Replicat process uses SQL to compile a SQL statement once and then executes it many times with different bind variables. You can configure the Replicat process so that it waits a specific amount of time before applying the replicated operations to the target database. For example, a delay may be desirable to prevent the propagation of errant SQL, to control data arrival across different time zones, or to allow time for other planned events to occur.

For the following two common uses cases of Oracle GoldenGate, the function of the Replicat process is as follows:

- **Initial Loads:** When you set up Oracle GoldenGate for initial loads, the Replicat process applies a static data copy to target objects or routes the data to a high-speed bulk-load utility.
- **Change Synchronization:** When you set up Oracle GoldenGate to keep the target database synchronized with the source database, the Replicat process applies the source operations to the target objects using a native database interface or ODBC, depending on the database type.

You can configure multiple Replicat processes with one or more Extract processes to increase the throughput. To preserve data integrity, each set of processes handles a different set of objects. To differentiate among Replicat processes, you assign each one a group name.

Types of Replicat

The Replicat process can be configured in the following three modes (also referred to as Replicat types):

- **Classic Replicat:** In classic mode, Replicat is a single-threaded process that uses standard SQL to apply data to the target tables. See [Classic Replicat](#) for more details.
- **Coordinated Replicat:** In this mode, the Replicat process is threaded. One coordinator thread spawns and coordinates one or more threads that execute replicated SQL operations in parallel. A coordinated Replicat process uses one parameter file and is monitored and managed as one unit. See [Coordinated Replicat](#) for more details.
- **Integrated Replicat:** In this mode, the Replicat process leverages the apply processing functionality that is available within the Oracle Database. Within a single Replicat configuration, multiple inbound server child processes known as apply servers apply transactions in parallel while preserving the original transaction atomicity. See [About Integrated Replicat](#) for more details.
- **Parallel Replicat:** Is a new variant of Replicat that applies transactions in parallel to improve performance. Parallel Replicat only supports replicating data from trails with full metadata, which requires the classic trail format. It takes into account dependencies between transactions, similar to Integrated Replicat. See [Parallel Replicat](#) for more details. Parallel Replicat is available in non-integrated (classic) and integrated mode.
- **Initial Load Replicat:** In this mode, when you set up Oracle GoldenGate for initial loads, the Replicat process applies a static data copy to target objects or routes the data to a high-speed bulk-load utility. See [Add Initial Load Extract Using the Admin Client](#) for more details.

About Classic or Non-Integrated Replicat

In classic mode, Replicat is a single-threaded process that uses standard SQL to apply data to the target tables. In this mode, Replicat operates as follows:

- Reads the Oracle GoldenGate trail.
- Performs data filtering, mapping, and conversion.
- Constructs SQL statements that represent source database DML or DDL transactions (in committed order).
- Applies the SQL to the target through the SQL interface that is supported for the given target database, such as ODBC or the native database interface.

As shown in this figure, you can apply transactions in parallel with a Classic Replicat, but only by partitioning the workload across multiple Replicat processes. A parameter file must be created for each Replicat.

To determine whether to use classic mode for any objects, you must determine whether the objects in one Replicat group will ever have dependencies on objects in any other Replicat group, transactional or otherwise. Not all workloads can be partitioned across multiple Replicat groups and still preserve the original transaction atomicity. For example, tables for which the workload routinely updates the primary key cannot easily be partitioned in this manner. DDL replication (if supported for the database) is not viable in this mode, nor is the use of some `SQLEXEC` or `EVENTACTIONS` features that base their actions on a specific record.

If your tables do not have any foreign key dependencies or updates to primary keys, classic mode may be suitable. Classic mode requires less overhead than coordinated mode.

About Coordinated Replicat

In coordinated mode, Replicat operates as follows:

- Reads the Oracle GoldenGate trail.
- Performs data filtering, mapping, and conversion.
- Performs data filtering, mapping, and conversion.
- Applies the SQL to the target through the SQL interface that is supported for the given target database, such as ODBC or the native database interface.

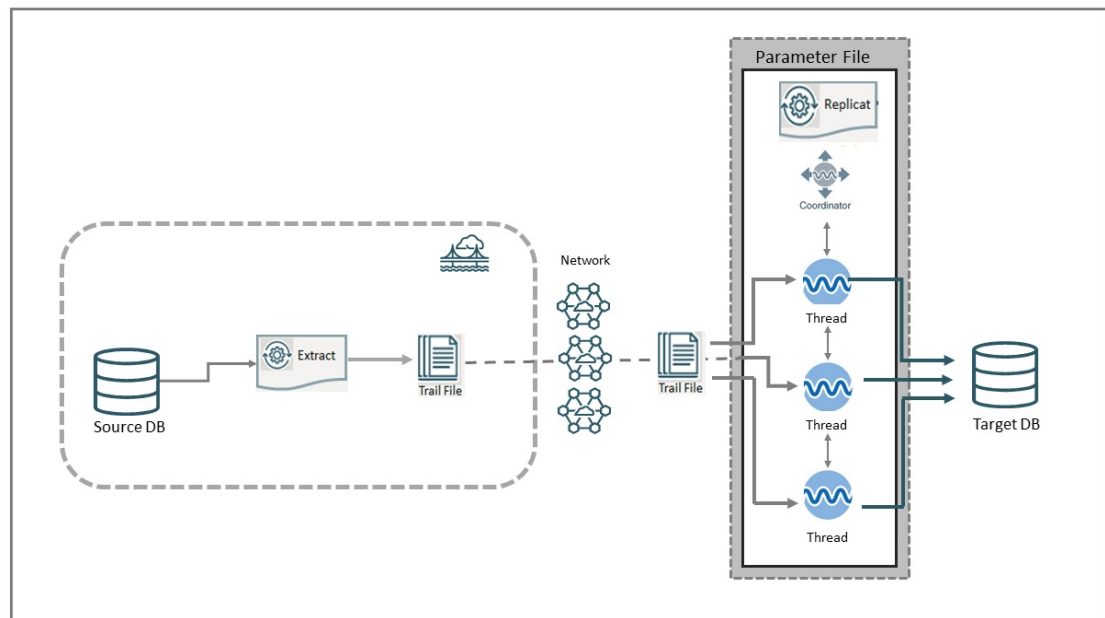
The difference between classic mode and coordinated mode is that Replicat is multi-threaded in coordinated mode. Within a single Replicat instance, multiple threads read the trail independently and apply transactions in parallel. Each thread handles the filtering, mapping, conversion, SQL construction, and error handling for its assigned workload. A coordinator thread coordinates the transactions across threads to account for dependencies among the threads.

The source transactions could be split across CR processes such that the integrity of the total source transaction is not maintained. The portion of the transaction processed by a CR process is done in committed order but the whole transaction across all CR processes is not.

Coordinated Replicat allows for user-defined partitioning of the workload so as to apply high volume transactions concurrently. In addition, it automatically coordinates the execution of transactions that require coordination, such as DDL, and primary key updates with `THREADRANGE` partitioning. Such a transaction is executed as one transaction in the target with full synchronization: it waits until all prior transactions are applied first, and all transactions after this barrier transaction have to wait until this barrier transaction is applied.

Only one parameter file is required for a coordinated Replicat, regardless of the number of threads. You use the `THREAD` or `THREADRANGE` option in the `MAP` statement to specify which threads process the transactions for those objects, and you specify the maximum number of threads when you create the Replicat group.

This figure illustrates the architecture of Coordinated Replicat.



As shown in this figure, the Coordinated Replicat includes the following two processes:

About Barrier Transactions

Barrier transactions are managed automatically in a coordinated Replicat configuration. Barrier transactions are transactions that require coordination across threads. Examples include DDL statements, transactions that include updates to primary keys, and certain `EVENTACTIONS` actions.

Optionally, you can force other transactions to be treated like a barrier transaction through the use of the `COORDINATED` keyword in a `MAP` statement. One use case for this would be force a `SQLEXEC` to be executed in a manner similar to a serial execution. This could be beneficial if the results can become ambiguous unless the state of the target is consistent across all transactions.

Note:

Coordinated Replicat doesn't do dependency calculations for non-barrier transactions when a mapped table is partitioned based on `THREADRANGE`. It relies on specified `THREADRANGE` columns to compute a hash value. It partitions the incoming data based on the hash value and sends all the records that match this hash value to same thread.

How Barrier Transactions are Processed

All threads converge and wait at the start of a barrier transaction. The barrier transaction is suspended until the other threads reach its start position. If any threads were already processing part of the barrier transaction, those threads perform a rollback. Grouped transactions, such as those controlled by the `BATCHSQL` or `GROUPTRANSOPS` parameters, are also rolled back and then reapplied until they reach the start of the barrier transaction.

All of the threads converge and wait at the start of the next transaction after the barrier transaction as well. The two synchronization points, before and after the barrier transaction, ensure that metadata operations and `EVENTACTIONS` actions all occur in the proper order relevant to the data operations.

Once the threads are synchronized at the start of the barrier transaction, the barrier transaction is processed serially by the thread that has the lowest thread ID among all of the threads specified in the MAP statements, and then parallel processing across threads is resumed. You can force barrier transactions to be processed through a specific thread, which is always thread 0, by specifying the `USEDEDICATEDCOORDINATIONTHREAD` parameter in the Replicat parameter file.

About Integrated Replicat

In integrated mode, Replicat leverages the apply processing functionality that is available within the target Oracle database. In this mode, Replicat reads the trail, constructs logical change records that represent source DML or DDL transactions, and transmits these records to an inbound server in the Oracle target database. The inbound server applies the data to the target database.



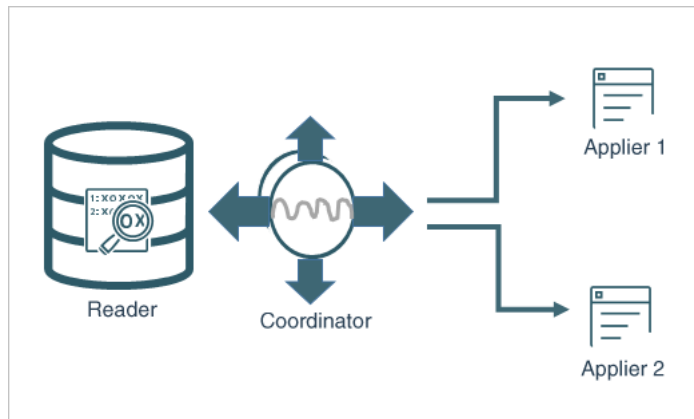
Note:

Integrated Replicat is an online process only. Do not use it to perform initial loads.

In integrated 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 transactions (in committed order). DDL is applied directly by Replicat.
- 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.

Within a single Replicat configuration, multiple inbound server child processes known as apply servers apply transactions in parallel while preserving the original transaction atomicity. You can increase this parallelism as much as your target system will support when you configure the Replicat process or dynamically as needed. The following diagram illustrates integrated Replicat configured with two parallel apply servers.



In the above diagram, Integrated Replicat applies transactions asynchronously. Transactions that do not have interdependencies can be safely executed and committed out of order to achieve fast throughput. Transactions with dependencies are guaranteed to be applied in the same order as on the source.

A reader process in the inbound server computes the dependencies among the transactions in the workload based on the constraints defined at the target database (primary key, unique, foreign key). Barrier transactions and DDL operations are managed automatically, as well. A coordinator process coordinates multiple transactions and maintains order among the apply servers.

If the inbound server does not support a configured feature or column type, Replicat disengages from the inbound server, waits for the inbound server to complete transactions in its queue, and then applies the transaction to the database in direct apply mode through OCI. Replicat resumes processing in integrated mode after applying the direct transaction.

The following features are applied in direct mode by Replicat:

- DDL operations
- Sequence operations
- `SQLEXEC` parameter within a `TABLE` or `MAP` parameter
- `EVENTACTIONS` processing

Because transactions are applied serially, heavy use of such operations may reduce the performance of the integrated Replicat mode. Integrated Replicat performs best when most of the apply processing can be performed in integrated mode.

 **Note:**

User exits are executed in integrated mode. However, user exit may produce unexpected results, if the exit code depends on data in the replication stream.

 **Note:**

Integrated Replicat requires that any foreign key columns are indexed.

Benefits of Integrated Replicat

The following are the benefits of using integrated Replicat versus non-integrated Replicat.

- Integrated Replicat enables heavy workloads to be partitioned automatically among parallel apply processes that apply multiple transactions concurrently, while preserving the integrity and atomicity of the source transaction. Both a minimum and maximum number of apply processes can be configured with the `PARALLELISM` and `MAX_PARALLELISM` parameters. Replicat automatically adds additional servers when the workload increases, and then adjusts downward again when the workload lightens.
- Integrated Replicat requires minimal work to configure. All work is configured within one Replicat parameter file, without configuring range partitions.
- High-performance apply streaming is enabled for integrated Replicat by means of a lightweight application programming interface (API) between Replicat and the inbound server.
- Barrier transactions are coordinated by integrated Replicat among multiple server apply processes.
- DDL operations are processed as direct transactions that force a barrier by waiting for server processing to complete before the DDL execution.
- Transient duplicate primary key updates are handled by integrated Replicat in a seamless manner.

Integrated Replicat Requirements

To use integrated Replicat, the following must be true.

- Supplemental logging must be enabled on the source database to support the computation of dependencies among tables and scheduling of concurrent transactions on the target. Instructions for enabling the required logging are in *Configuring Logging Properties*. This logging can be enabled at any time up to, but before you start the Oracle GoldenGate processes.
- Integrated Parallel Replicat is supported on Oracle Database 12.2.0.1 and greater.

About Parallel Replicat

Parallel Replicat is another variant of Replicat that applies transactions in parallel to improve performance.

It takes into account dependencies between transactions, similar to Integrated Replicat. The dependency computation, parallelism of the mapping and apply is performed outside the database so can be off-loaded to another server. The transaction integrity is maintained in this process. In addition, parallel Replicat supports the parallel apply of large transactions by splitting a large transaction into chunks and applying them in parallel.

Parallel Replicat supports the following two modes: Integrated and Non-integrated. Only Oracle database supports parallel Replicat and integrated parallel Replicat. However, parallel Replicat supports all databases when using the non-integrated option.

To use parallel Replicat, you need to ensure that you have the following values, which are also the default values:

- Metadata in the trail (which means you can't use parallel Replicat if your trails are formatted below 12.1).

- You must have scheduling columns in your trail file.
- You must use `UPDATERCORDFORMAT COMPACT`.

With integrated parallel Replicat, the Replicat sends the LCRs to the inbound server, which applies the data to the target database, and in regular parallel Replicat, Oracle GoldenGate applies the LCR as a SQL statement directly to the database, similar to how the other non-integrated Replicats work.

 **Note:**

For best performance for an OLTP workload, parallel Replicat in non-integrated mode is recommended.

The components of parallel Replicat are:

- Mappers operate in parallel to read the trail, map trail records, convert the mapped records to the Integrated Replicat LCR format, and send the LCRs to the Merger for further processing. While one Mapper maps one set of transactions, the next Mapper maps the next set of transactions. The trail information is split and the trail file is untouched because it orders trail information in order.
- Master processes have two threads, Collater and Scheduler. The Collater receives mapped transactions from the Mappers and puts them back into trail order for dependency calculation. The Scheduler calculates dependencies between transactions, groups transactions into independent batches, and sends the batches to the Appliers to be applied to the target database.
- Appliers reorder records within a batch for array execution. It applies the batch to the target database and performs error handling. It also tracks applied transactions in checkpoint tables.

 **Note:**

Parallel Replicat requires that any foreign key columns are indexed.

Benefits of Parallel Replicat

The following are the benefits of using parallel Replicat:

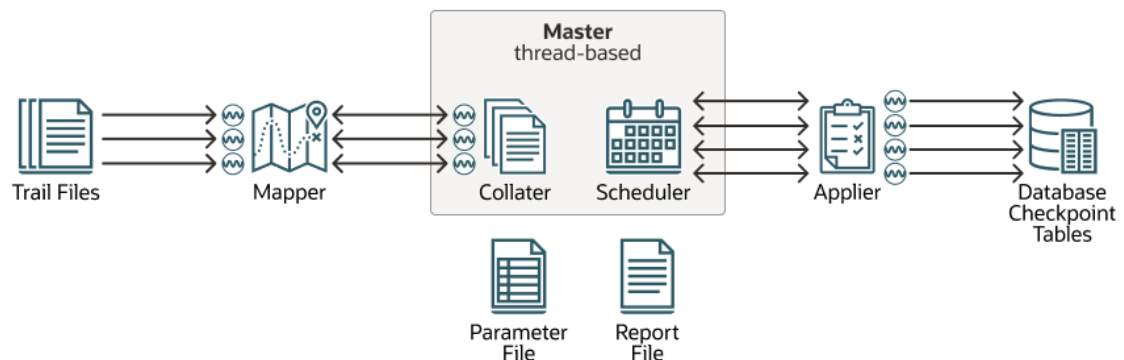
- Integrated Parallel Replicat enables heavy workloads to be partitioned automatically among parallel apply processes that apply multiple transactions concurrently, while preserving the integrity and atomicity of the source transaction. Both a minimum and maximum number of apply processes can be configured with the `PARALLELISM` and `MAX_PARALLELISM` parameters. Replicat automatically adds additional servers when the workload increases, and then adjusts downward again when the workload lightens.
- Integrated Parallel Replicat requires minimal work to configure. All work is configured within one Replicat parameter file, without configuring range partitions.
- High-performance apply streaming is enabled for integrated parallel Replicat by means of a lightweight application programming interface (API) between Replicat and the inbound server.

- Barrier transactions are coordinated by integrated parallel Replicat among multiple server apply processes.
- DDL operations are processed as direct transactions that force a barrier by waiting for server processing to complete before the DDL execution.
- Transient duplicate primary key updates are handled by integrated parallel Replicat in a seamless manner.
- Parallel Replicat can break a single large transaction into smaller chunks and apply those chunks in parallel. See `SPLIT_TRANS_RECS` for details.

Parallel Replication Architecture

Parallel replication processes leverage the apply processing functionality that is available within the Oracle Database in integrated mode. Within a single Replicat configuration, multiple inbound server child processes, known as apply servers, apply transactions in parallel while preserving the original transaction atomicity.

The following architecture diagram depicts the flow of change records through the various processes of a parallel replication from the trail files to the target database, for a non-integrated parallel Replicat.



The following is the description of the architecture diagram given above:

- The Mappers read the trail file and map records, forward the mapped records to the Master. The batches are sent to the Appliers where they are applied to the target database.
- The Master process consists of two separate threads, Collater and Scheduler. The Collater is responsible for managing and communicating with the Mappers, along with receiving the mapped transactions and reordering them into a single in-order stream. The Scheduler is responsible for managing and communicating with the Appliers, along with reading transactions from the Collater, batching them, and scheduling them to Appliers.
- The Scheduler controller communicates with the Scheduler to gather any necessary information (such as, the current low watermark position). The Scheduler controller is required for CDB mode for Oracle Database because it is responsible for aggregating information pertaining to the different target PDBs and reporting a unified picture. The Scheduler controller is created for simplicity and uniformity of implementation, even when not in CDB mode. Every process reads the parameter file and shares a single checkpoint file.

Basic Parameters for Parallel Replicat

The following table lists the basic parallel Replicat parameters and their description.

Parameter	Description
MAP_PARALLELISM	Configures 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.
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.
MIN_APPLY_PARALLELISM MAX_APPLY_PARALLELISM	The 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 same time.
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.
COMMIT_SERIALIZATION	Enables commit FULL serialization mode, which forces transactions to be committed in trail order.
Advanced Parameters	
LOOK_AHEAD_TRANSACTIONS	Controls how far ahead the Scheduler looks when batching transactions. The default value is 10000.
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.

Example Parameter File

```

REPLICAT repe USERID ggadmin, password ***
MAP_PARALLELISM 3
MIN_APPLY_PARALLELISM 2
MAX_APPLY_PARALLELISM 10
SPLIT_TRANS_RECS 60000
MAP *.* , TARGET *.*;

```

Select a Replicat Type for the Deployment


Replicat is responsible for applying trail data to the target database. Although you can choose from different types of Replicat modes, Oracle recommends that you use the parallel nonintegrated Replicat, unless a specific feature requires a different type of Replicat. Parallel Replicat is available for both Oracle and non-Oracle databases.

The following table lists the features supported by the respective Replicats.

Feature	Parallel Replicat	Integrated Replicat	Coordinated Replicat	Classic Replicat
Batch Processing	Yes	Yes	Yes	Yes

Feature	Parallel Replicat	Integrated Replicat	Coordinated Replicat	Classic Replicat
Barrier Transactions	Yes	Yes	Yes	No
Dependency Computation	Yes	Yes	No	No

Feature	Parallel Replicat	Integrated Replicat	Coordinated Replicat	Classic Replicat
Auto-parallelism	Yes	Yes	No	No

 **Note:**

Auto-parallelism is disabled by default. Only you

Feature	Parallel Replicat	Integrated Replicat	Coordinated Replicat	Classic Replicat
	rt th r e a d s a r e u s e d i n t h e d e f a u l t s e t t i n g s . I f y o u w a n t t o c h a n g e R e			

Feature	Parallel Replicat	Integrated Replicat	Coordinated Replicat	Classic Replicat
	<div><div></div><div>p l i c a t t o u s e M I N — P A R A L L E L I S M a n d M A X — P A R A L L E L I S M , t h e n a u t</div></div>			

Feature	Parallel Replicat	Integrated Replicat	Coordinated Replicat	Classic Replicat
<div> o - p a r a l l e l i s m i s u s e d . </div>				
DML Handler	Yes, Integrated mode	Yes	No	No
Procedural Replication	Yes, used for integrated Parallel Replicat (iPR)	Yes	No	No
Auto CDR	Yes, used by iPR only	Yes	No	No
Dependency-aware Transaction Split	Yes	No	No	No
Cross-RAC-node Processing	Yes	No	Yes	No
ALLOWDUPTARGETMAP See ALLOWDUPTARGETMAP NOALLOWDUPTARGETMAP	No, Oracle Database with iPR	No, Oracle Database	Yes	Yes

Add a Replicat

Use the Administration Server to add Replicats. Before adding a Replicat, make sure that you have completed created the checkpoint and heartbeat tables. See [Before Adding Extract and Replicat Processes](#).

The following steps describe the creation of a non-integrated Parallel Replicat. You can choose to set up a different type of Replicat using the same steps, however, the Replicat parameters may differ.

1. Click the + sign next to Replicats on the Administration Server home page.
The Add Replicat page is displayed.
2. Select a Replicat type and click **Next**.
The types of Replicat are:
 - Integrated Replicat
 - Nonintegrated Replicat
 - Coordinated Replicat
 - Parallel Replicat: If you select this option, then select an integrated or nonintegrated parallel Replicat.
3. Enter the required Replicat options on the Replicat Options page and click **Next**. To know more about the Replicat options, see the online help.
4. Enter the required information making sure that you complete the Credential Domain and Credential Alias fields before completing the Checkpoint Table field, and then select your newly created Checkpoint Table from the list.
5. For managed processes, the options to enter are:

Option	Description	Extract Type
Intent	What you want the Extract to be used for, such as High Availability or the Unidirectional default.	Classic, Integrated, and Initial Load
Begin	How you want the Extract to start. At a custom time that you select, a database CSN, or the Now default.	Classic and Integrated
Trail Name	A two character trail name.	Classic and Integrated
Trail Subdirectory, Size, Sequence, and Offset	You can further configure the trail details.	Classic and Integrated
Remote	Set if the trail is not on the same server.	Classic and Integrated
Thread Number	Set to a specific redo log number. The default is 1.	Classic
Encryption Profile	Provide the name of the encryption profile for the Extract. If no encryption profile is created, then the default encryption profile is selected, by default	Classic, Integrated, and Initial Load.
Encryption Profile Type	Provide the type of Key Management Service being used. Oracle Key Vault is selected by default.	Classic, Integrated, and Initial Load.
Managed Options	X	X
Profile Name	Provides the name of the autostart and autorestart profile. You can select the default or custom options.	Classic, Integrated, and Initial Load.

Option	Description	Extract Type
Critical to deployment health	Enable this option if the profile is critical for the deployment health.	Classic, Integrated, and Initial Load.
Auto Start	Enables autostart for the process.	Enables autostart for the process.
Max Retries	Specify the maximum number of retries to try to start the process	Classic, Integrated, and Initial Load.
Retry Delay	Delay time in trying to start the process	Classic, Integrated, and Initial Load.
Retries Window	The duration interval to try to start the process	Classic, Integrated, and Initial Load.
Restart on Failure only	If true the task is only restarted if it failes	Classic, Integrated, and Initial Load.
Disable Task After Retries Exhausted	If true then the task is disabled after exhausting all attempts to restart the process.	Classic, Integrated, and Initial Load.

6. Click **Create and Run** to create and run the Replicat.

Basic Parameters for Parallel Replicat

The following table lists the basic parallel Replicat parameters and their description.

Parameter	Description
MAP_PARALLELISM	Configures 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.
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 4.
MIN_APPLY_PARALLELISM MAX_APPLY_PARALLELISM	The 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 <i>not</i> use with APPLY_PARALLELISM at the same time.
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.
COMMIT_SERIALIZATION	Enables commit FULL serialization mode, which forces transactions to be committed in trail order.
Advanced Parameters	
LOOK_AHEAD_TRANSACTIONS	Controls how far ahead the Scheduler looks when batching transactions. The default value is 10000.

Parameter	Description
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.

Example Parameter File

```
replicat repA
userid ggadmin, password ***
MAP_PARALLELISM 3
MIN_APPLY_PARALLELISM 2
MAX_APPLY_PARALLELISM 10
SPLIT_TRANS_RECS 1000
map *.* , target *.*;
```

Additional Parameters for Integrated Replicat

You can set these parameters by using the `DBOPTIONS` parameter with the `INTEGRATEDPARAMS` option or dynamically by issuing the `SEND REPLICAT` command with the `INTEGRATEDPARAMS` option from the command line.

The default Replicat configuration should be sufficient. However, if needed, you can set the following inbound server parameters to support specific requirements.



Note:

For detailed information and usage guidance for these parameters, see the `DBMS_APPLY_ADM` section in *Oracle Database PL/SQL Packages and Types Reference*.

See *Parameters and Functions Reference for Oracle GoldenGate* for more information about the `DBOPTIONS` parameter.

- COMMIT_SERIALIZATION:** Controls the order in which applied transactions are committed and has 2 modes, `DEPENDENT_TRANSACTIONS` and `FULL`. The default mode for Oracle GoldenGate is `DEPENDENT_TRANSACTIONS` where dependent transactions are applied in the correct order though may not necessarily be applied in source commit order. In `FULL` mode, the source commit order is enforced when applying transactions.
- BATCHSQL_MODE:** Controls the batch execution scheduling mode including pending dependencies. A pending dependency is a dependency on another transaction that has already been scheduled, but not completely executed. The default is `DEPENDENT`. You can use following three modes:

DEPENDENT

Dependency aware scheduling without an early start. Batched transactions are scheduled when there are no pending dependencies.

DEPENDENT_EAGER

Dependency aware batching with early start. Batched transactions are scheduled irrespective of pending dependencies.

SEQUENTIAL

Sequential batching. Transactions are batched by grouping the transactions sequentially based on the original commit order.

- **DISABLE_ON_ERROR:** Determines whether the apply server is disabled or continues on an unresolved error. The default for Oracle GoldenGate is **N** (continue on errors), however, you can set the option to **Y** if you need to disable the apply server when an error occurs.
- **EAGER_SIZE:** Sets a threshold for the size of a transaction (in number of LCRs) after which Oracle GoldenGate starts applying data before the commit record is received. The default for Oracle GoldenGate is 15100.
- **ENABLE_XSTREAM_TABLE_STATS:** Controls whether statistics on applied transactions are recorded in the **V\$GOLDENGATE_TABLE_STATS** view or not collected at all. The default for Oracle GoldenGate is **Y** (collect statistics).
- **MAX_PARALLELISM:** Limits the number of apply servers that can be used when the load is heavy. This number is reduced again when the workload subsides. The automatic tuning of the number of apply servers is effective only if **PARALLELISM** is greater than 1 and **MAX_PARALLELISM** is greater than **PARALLELISM**. If **PARALLELISM** is equal to **MAX_PARALLELISM**, the number of apply servers remains constant during the workload. The default for Oracle GoldenGate is 50.
- **MAX_SGA_SIZE:** Controls the amount of shared memory used by the inbound server. The shared memory is obtained from the streams pool of the SGA. The default for Oracle GoldenGate is **INFINITE**.
- **MESSAGE_TRACKING_FREQUENCY:** Controls how often LCRs are marked for high-level LCR tracing through the apply processing. The default value is 2000000, meaning that every 2 millionth LCR is traced. A value of zero (0) disables LCR tracing.
- **PARALLELISM:** Sets a minimum number of apply servers that can be used under normal conditions. Setting **PARALLELISM** to 1 disables apply parallelism, and transactions are applied with a single apply server process. The default for Oracle GoldenGate is 4. For Oracle Standard Edition, this must be set to 1.
- **PARALLELISM_INTERVAL:** Sets the interval in seconds at which the current workload activity is computed. Replicat calculates the mean throughput every 5 X **PARALLELISM_INTERVAL** seconds. After each calculation, the apply component can increase or decrease the number of apply servers to try to improve throughput. If throughput is improved, the apply component keeps the new number of apply servers. The parallelism interval is used only if **PARALLELISM** is set to a value greater than one and the **MAX_PARALLELISM** value is greater than the **PARALLELISM** value. The default is 5 seconds.
- **PRESERVE_ENCRYPTION:** Controls whether to preserve encryption for columns encrypted using Transparent Data Encryption. The default for Oracle GoldenGate is **N** (do not apply the data in encrypted form).
- **OPTIMIZE_PROGRESS_TABLE:** Integrated Delivery uses this table to track the transactions that have been applied. It is used for duplicate avoidance in the event of failure or restart. If it is set to **N** (the default), then the progress table is updated synchronously with the apply of each replicated transaction. When set to **Y**, rather than populating the progress table synchronously, markers are dropped into the redo stream so when the apply process starts up, it mines the redo logs for these markers, and then updates the progress table for the previously applied transactions.

- **TRACE_LEVEL:** Controls the level of tracing for the Replicat inbound server. For use only with guidance from Oracle Support. The default for Oracle GoldenGate is 0 (no tracing).
- **WRITE_ALERT_LOG:** Controls whether the Replicat inbound server writes messages to the Oracle alert log. The default for Oracle GoldenGate is Y (yes).

Example: Add a Nonintegrated Parallel Replicat Using Admin Client

You can create a parallel Replicat from the user interface or the command line interface.

Before you start creating the parallel Replicat, make sure that you've select the checkpoint table.

1. Go the `bin` directory of your Oracle GoldenGatehome directory.

```
cd $OGG_HOME/bin
```

2. Start the Admin Client.

```
adminclient
```

The Admin Client command prompt is displayed.

```
OGG (not connected) 12>
```

3. Connect to the Service Manager deployment source:

```
connect https://localhost:9500 deployment Target1 as oggadmin password welcome1
```

You must use `http` or `https` in the connection string; this example is a non-SSL connection.

4. Add the Parallel Replicat, which may take a few minutes to complete:

```
add replicat R1, parallel, exttrail bb checkpointtable ggadmin.ggcheckpoint
```

You could use just the two character trail name as part of the `ADD REPLICAT` or you can use the full path, such as `/u01/oggdeployments/target1/var/lib/data/bb`.

5. Verify that the Replicat is running:

```
info replicat R1
```

Messages similar to the following are displayed:

```
REPLICAT   R1           Initialized   2016-12-20 13:56   Status RUNNING
NONINTEGRATED
Parallel
Checkpoint Lag      00:00:00 (updated 00:00:22 ago)
Process ID          30007
Log Read
Checkpoint File ./ra000000000First Record RBA 0
```

Using Replicat Actions

Various Replicat actions can be performed from the Administration Server Overview page.

You can change the status of the Replicat process using the Actions button to:

Action	Result
Details	<p>Displays the Process Information page that has the following details:</p> <ul style="list-style-type: none"> • Statistics: Displays the active replication maps along with replication statistics based on the type of Replicat. • Parameters: Displays the parameters configured when the Replicat was added. You can change these parameters to adjust your Replicat. • Report: Displays the details about the Replicat including the parameters with which the replicat is running, and run time messages. • Checkpoint: Displays the checkpoint log name, path, timestamp, sequence, and offset value. You can click the Checkpoint Detail icon to view elaborate information about the checkpoint.
Start/Stop	The Replicat starts or stops immediately.
Start/Stop (in the background)	The Replicat is started or stopped using a background process.
Start with Options	Allows you to change the Replicat start point, CSN, filter duplicates, and threads options, then starts the Replicat.
Force Stop	The Replicat is immediately, forcibly stopped.
Alter	Allows you to change when the Replicat begins, the description, and the intent. It does not start the Replicat.
Delete	Deletes the Replicat if you confirm the deletion.

When you change the status, the list options change accordingly. As status are changing, the icons change to indicate the current and final status. The events are added to the Critical Events table. Additionally, progress pop-up messages appear in the bottom of your browser.

Review Critical Events

You can review and search for critical events from the Administration Server home page, once you set up the distribution path.

Once you set up the Extracts and Replicats along with the Distribution path, you are able to see the critical events associated with them.

Search for Critical Events from the Review Critical Events Table

The Review Critical Events table displays the severity, error code, and error messages for critical events. You can view 20 error messages on a single page and you can also search for specific events.

Additionally, you can examine events in depth from the Performance Metrics Server. For details see [Monitor Processes from the Performance Metrics Server](#)

8

Instantiate

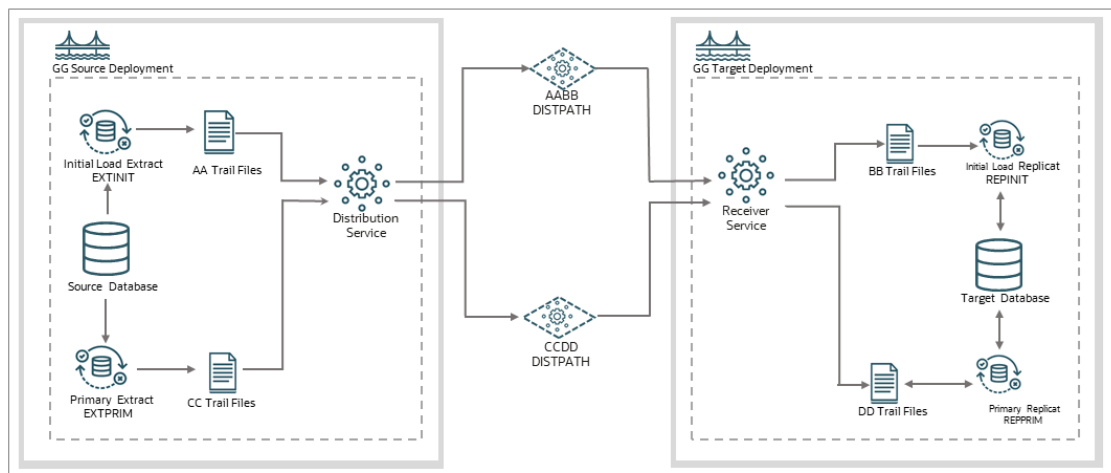
This section lists details about instantiating with Initial Load Extract and adding the Initial Load Extract using the Admin Client.

About Instantiating with Initial Load Extract

Using the initial load Extract for instantiation, you can replicate data precisely from a source to a target database with zero data loss. To configure this Extract, you'll require a combination of file-based initial load and change data capture (CDC) processes.

In Microservices Architecture, the process of instantiation includes the following tasks:

- Add and configure an Initial Load Extract: This Extract is used to copy the existing contents of one or more tables from the source to the target database.
- Configure Change Data Capture: Used to copy transactional changes from the source to the target database.



Note:

MA doesn't support loading data with an Oracle GoldenGate direct load.

File-based initial load process is the preferred method for performing data replication in MA. Its key components are:

- Initial Load Extract and Replicat: Replicates the existing content of the database tables.
- Primary Extract and Replicat: Replicates change data from the database tables.
- Distribution Paths: Transfers trail files to the target system.

Before you begin, make sure that the database credential alias is created.

You can use the Oracle GoldenGate web interface, Admin Client, or cURL commands to set up this configuration.

Add Initial Load Extract Using the Admin Client

Learn about adding the Initial Load Extract using the Admin Client.

Step 1: Create a Primary Extract

Precise instantiation is used to replicate database resources correctly from the source to the target database. The primary Extract is started first to initiate change data capture early. Precise instantiation is based on the following assumptions:



Note:

For precise instantiation to work, the instantiation SCN must come after the registration SCN.

- The primary Extract is started. It is responsible for change data capture and noting its registration SCN.
- The database is monitored. The database waits for the oldest open transaction's SCN to come after the registration SCN. This is the instantiation SCN.
- The instantiation SCN is used when creating the initial load Extract and Replicat processes.
- The instantiation SCN is used to create the primary Replicat, once the initial load replication is complete.
- For MySQL, precise instantiation is applicable only for MySQL source and target databases, and is implemented using the `Dump` utility of the MySQL shell. For more information on the `Dump` utility, see [MySQL Dump Utility](#).

To begin, create and start the primary Extract `EXTPRIM` from the AdminClient, as shown in the following example:

Command:

```
OGG (not connected) 1> CONNECT https://oggdep.example.com:9100 as oggadmin  
password oggadmin !
```

Output:

```
Using default deployment 'OGGDEP'
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP) 2> DBLOGIN USERIDALIAS oggadmin
```

Output:

```
Successfully logged into database.
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP) 3> ADD EXTRACT extprim INTEGRATED  
TRANLOG BEGIN NOW
```

Output:

```
2018-03-16T13:37:07Z INFO OGG-08100 EXTRACT (Integrated) added.
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 4> REGISTER EXTRACT  
extprim DATABASE
```

Output:

```
2018-03-16T13:37:30Z INFO OGG-02003 Extract EXTPRIM successfully registered with  
database at SCN 1608891.
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 5> EDIT PARAMS extprim
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 6> VIEW PARAMS extprim
```

Output:

```
--  
-- E X T P R I M . p r m  
-- Primary Extract Parameter File  
--  
Extract EXTPRIM  
UseridAlias oggadmin  
ExtTrail AA  
Table user01.*;
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 7> ADD EXTTRAIL aa  
EXTRACT extprim
```

Output:

```
2018-03-16T13:37:55Z INFO OGG-08100 EXTTRAIL added.
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 8> START EXTRACT extprim
```

Output:

```
2018-03-16T13:38:02Z INFO OGG-00975 EXTRACT EXTPRIM starting  
2018-03-16T13:38:02Z INFO OGG-15426 EXTRACT EXTPRIM started
```

In this example, oggadmin is the database credential alias.

After creating the primary Extract, retrieve the SCN registration number. Run the REGISTER EXTRACT command in the AdminClient. The following example retrieves an SCN value of 1608891.

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 4> REGISTER EXTRACT  
extprim DATABASE
```

Output:

```
2018-03-16T13:37:30Z INFO OGG-02003 Extract EXTPRIM successfully registered with  
database at SCN 1608891.
```

Step 2: Determine the Instantiation SCN

The Administration Service in Oracle GoldenGate Microservices Architecture, provides an endpoint for retrieving information about open database transactions. This information can be used to identify the SCN to use when instantiating the initial load Extract.

In the following example, the instantiation SCN is 1609723, which is the oldest SCN of all open transactions that is also past the registration SCN of 1608891, identified in the previous step.

```
-- Query for active transactions  
--  
SELECT T.START_SCN, T.STATUS TSTATUS, T.START_DATE,  
       S.SID, S.SERIAL#, S.INST_ID, S.USERNAME, S.OSUSER, S.STATUS SSTATUS,  
       S.LOGON_TIME  
  FROM gv$transaction T  
 INNER JOIN gv$session S  
    ON s.saddr = t.ses_addr  
  
UNION ALL  
  
--  
-- Query for current status  
--  
SELECT CURRENT_SCN, 'CURRENT', CURRENT_DATE,  
       NULL, NULL, NULL, 'SYS', NULL, NULL, NULL  
  from v$database  
  
ORDER BY 1;
```

The results of this query can be used to determine the instantiation SCN. The results for this specific query are:

```
1538916 ACTIVE 2018-03-16 18:10:31.0 3865 9176 1 OGGADMIN oracle INACTIVE  
2018-03-16 18:10:26.0 1540555 CURRENT 2018-03-16 18:21:50.0 SYS
```

The SCN used to instantiate the initial load Extract is obtained using SQL*Plus. In the following example, the SQL query uses the instantiation SCN value as 1624963, which is the oldest SCN of all open transactions that are also past the registration SCN of 1608891.

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 14> SHELL ECHO  
'SELECT MIN(START_SCN) FROM gv$transaction;' | ${ORACLE_HOME}/bin/sqlplus -  
S / as sysdba  
  
MIN(START_SCN)  
-----  
1624963
```

If there are no open transactions, then this SQL query returns an empty result. A detailed query that takes into account the situation where there are no open transactions is:

```
SELECT MIN(SCN) as INSTANTIATION_SCN
FROM (SELECT MIN(START_SCN) as SCN
      FROM gv$transaction
      UNION ALL
      SELECT CURRENT_SCN
      FROM gv$database);
```

Step 3: Create and Start the Initial Load Replicat

Before you begin this step, make sure that the checkpoint table `oggadmin.checkpoints`, already exists on the target system. The initial load Replicat is responsible for populating the target database. Run the following command on the AdminClient to create and start the initial load Replicat (REPINIT):

Command:

```
OGG (not connected) 1> CONNECT https://oggdep.example.com:9100 as oggadmin
password oggadmin !
```

Output:

```
Using default deployment 'OGGDEP'
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP) 2> DBLOGIN USERIDALIAS oggadmin
```

Output:

```
Successfully logged into database.
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 3> ADD CHECKPOINTTABLE
oggadmin.checkpoints
```

Output:

```
ADD "oggadmin.checkpoints" succeeded.
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 4> ADD REPLICAT repinit
EXTTRAIL dd CHECKPOINTTABLE oggadmin.checkpoints
```

Output:

```
2018-03-16T13:56:41Z INFO OGG-08100 REPLICAT added.
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 5> EDIT PARAMS repinit
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 6> VIEW PARAMS repinit
```

Output:

```
--
--  R E P I N I T . p r m
--  File-Based Initial Load Replicat Parameter File
--
Replicat      REPINIT
UseridAlias   oggadmin
Map           user01.*
  Target      user01.*;
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 7> START REPLICAT
repinit
```

Output:

```
2018-03-16T13:58:21Z  INFO      OGG-00975  REPLICAT REPINIT starting
2018-03-16T13:58:21Z  INFO      OGG-15426  REPLICAT REPINIT started
```

Step 4: Create and start the Initial Load Extract

Using the instantiation SCN that you retrieved (1624963), the initial load Extract is created to write contents of the database tables to the trail. Create and start the initial load extract, EXTINIT.

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 15> ADD EXTRACT extinit
SOURCEISTABLE sourceistable
```

Output:

```
2018-03-16T14:08:38Z  INFO      OGG-08100  EXTRACT added.
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 16> EDIT PARAMS extinit
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 17> VIEW PARAMS extinit
```

Output:

```
--
--  E X T I N I T . p r m
--  File-Based Initial Load Extract Parameter File
--
Extract       EXTINIT
UseridAlias   oggadmin
ExtFile       CC Megabytes 2000 Purge
Table         user01.*, SQLPredicate "As Of SCN 1609723";
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 18> START EXTRACT  
extinit
```

Output:

```
2018-03-16T14:13:42Z INFO OGG-00975 EXTRACT EXTINIT starting  
2018-03-16T14:13:42Z INFO OGG-15426 EXTRACT EXTINIT started
```

Step 5: Create the Distribution Paths

Create two distribution paths (AABB and CCDD) for copying the local trails to the remote host from the Admin Client:

Command:

```
OGG (https://oggdep.example.com:9100 oggdep) 15> ADD DISTPATH aabb SOURCE  
TRAIL://oggdep.example.com:9102/services/v2/sources?trail=AA target wss://  
dallas.ogudevops.us:9103/services/v2/targets?trail=BB
```

Output:

```
2018-03-16T17:28:27Z INFO OGG-08511 The path 'AABB' has been added.
```

Command:

```
OGG (https://oggdep.ogudevops.us:9100 oggdep) 16> ADD DISTPATH ccdd SOURCE  
TRAIL://oggdep.example.com:9102/services/v2/sources?trail=CC target wss://  
dallas.ogudevops.us:9103/services/v2/targets?trail=DD
```

Output:

```
2018-03-16T17:28:35Z INFO OGG-08511 The path 'CCDD' has been added.
```

Command:

```
OGG (https://oggdep.example.com:9100 oggdep) 17> START DISTPATH aabb
```

Output:

```
2018-03-16T17:28:42Z INFO OGG-08513 The path 'AABB' has been started.
```

Command:

```
OGG (https://oggdep.example.com:9100 oggdep) 18> START DISTPATH ccdd
```

Output:

```
2018-03-16T17:28:47Z INFO OGG-08513 The path 'CCDD' has been started.
```

If you use the `ogg` protocol instead of `wss`, then you must use the `TARGETTYPE` option. The syntax in that case would be:

```
ADD DISTPATH path-name SOURCE source-uri TARGET target-uri [ TARGETTYPE ( MANAGER  
| COLLECTOR | RECVSRVR ) ]
```

`TARGETTYPE` specifies the target type in case the distribution path uses the legacy protocol. This argument is only valid if the target URI schema is `ogg`.

Step 6: Create the Primary Replicat

Once the initial load Extract and Replicat complete, they can be deleted. Then, the primary Replicat process is created on the remote host for applying change data to the target database.

Use the AdminClient to create the primary Replicat process.

**Note:**

The primary Replicat is started at the instantiation SCN.

Command:

```
OGG (https://oggdep.example.com:9100 oggdep as oggadmin) 12> ADD REPLICAT repprim  
EXTTRAIL bb CHECKPOINTTABLE oggadmin.checkpoints
```

Output:

```
2018-03-16T17:37:46Z INFO OGG-08100 REPLICAT added.
```

Command: EDIT PARAMS

```
OGG (https://oggdep.example.com:9100 oggdep as oggadmin) 13> EDIT PARAMS repprim
```

Command:

```
OGG (https://oggdep.example.com:9100 oggdep as oggadmin) 14> VIEW PARAMS repprim
```

Output:

```
--  
-- R E P P R I M . p r m  
-- Replicat Parameter File  
--  
Replicat      REPPRIM  
USERIDALIAS  oggadmin  
Map           user01.*  
  Target      user01.*;
```

Command:

```
OGG (https://oggdep.example.com:9100 oggdep as oggadmin) 15> START REPLICAT  
repprim ATCSN 1624963
```

Output:

```
2018-03-16T17:38:10Z INFO      OGG-00975  REPLICAT REPPRIM starting  
2018-03-16T17:38:10Z INFO      OGG-15426  REPLICAT REPPRIM started
```

9

Administer

Learn about Microservices command line interface, parameters files, bi-directional configuration, procedural replication, automatic and manual conflict detection and resolution, mapping and manipulating data, and handling processing errors.

Data Management

Learn about various aspects of data management in Oracle GoldenGate, including DDL and DML replication, requirements and steps for configuring procedural replication, using SQLEXEC, Event Actions, and User Exits.

Oracle: DDL Replication

Learn about DDL replication in Oracle.

Extract supports the DDL capture method for Oracle 11.2.0.4 or later. An Extract can capture DDL operations from a source Oracle database natively through the Oracle logmining server.

Prerequisites for Configuring DDL

Oracle databases that have the `COMPATIBLE` parameter set to 11.2.0.4 or higher support DDL capture through the database logmining server. This method is known as native DDL capture. Native DDL capture is the only supported method for capturing DDL from a multitenant container database.

For downstream mining, the source database must also have database `COMPATIBLE` set to 11.2.0.4 or higher to support DDL capture through the database logmining server.

Overview of DDL Synchronization

Oracle GoldenGate supports the synchronization of DDL operations from one database to another.

DDL synchronization can be active when:

- business applications are actively accessing and updating the source and target objects.
- Oracle GoldenGate transactional data synchronization is active.

The components that support the replication of DDL and the replication of transactional data changes (DML) are independent of each other. Therefore, you can synchronize:

- DDL changes
- DML changes
- DDL and DML

For a list of supported objects and operations for DDL support for Oracle, see [Details of Support for Objects and Operations in Oracle DDL](#).

Limitations of Oracle GoldenGate DDL Support

Here are the limitations of Oracle GoldenGate DDL support.

For any additional details that were included after this documentation was published, see the *Release Notes for Oracle GoldenGate*.

DDL Statement Length

Oracle GoldenGate measures the length of a DDL statement in bytes, not in characters. The supported length is approximately 4 MB, allowing for some internal overhead that can vary in size depending on the name of the affected object and its DDL type, among other characteristics. If the DDL is longer than the supported size, Extract will issue a warning and ignore the DDL operation.

If Extract is capturing DDL by means of the DDL trigger, the ignored DDL is saved in the marker table. You can capture Oracle DDL statements that are ignored, as well as any other Oracle DDL statement, by using the `ddl_ddl2file.sql` script, which saves the DDL operation to a text file in the `USER_DUMP_DEST` directory of Oracle. The script prompts for the following input:

- The name of the schema that contains the Oracle GoldenGate DDL objects, which is specified in the `GLOBALS` file.
- The Oracle GoldenGate marker sequence number, which is recorded in the Extract report file when `DDLOPTIONS` with the `REPORT` option is used in the Extract parameter file.
- A name for the output file.

Supported Topologies

Oracle GoldenGate supports DDL synchronization only in a like-to-like configuration. The source and target object definitions must be identical.

DDL replication is only supported for Oracle to Oracle replication. It is not supported between different databases, like Oracle to Teradata, or SQL Server to Oracle. Oracle GoldenGate does not support DDL on a standby database. Oracle GoldenGate supports DDL replication in all supported unidirectional configurations, and in bidirectional configurations between two, and only two, systems.

Filtering, Mapping, and Transformation

DDL operations cannot be transformed by any Oracle GoldenGate process. However, source DDL can be mapped and filtered to a different target object by a primary Extract or a Replicat process using `DDL INCLUDE` and `EXCLUDE` options in the Extract and Replicat parameter files.

For details about using DDL filtering, mapping, and transformation, see DDL.

Renames

`RENAME` operations on tables are converted to the equivalent `ALTER TABLE RENAME` so that a schema name can be included in the target DDL statement. For example `RENAME EMP TO EMPLOYEES` could be changed to `ALTER TABLE hr.EMP RENAME TO hr.EMPLOYEES`.

The conversion is reported in the Replicat process report file.

Interactions Between Fetches from a Table and DDL

Oracle GoldenGate supports some data types by identifying the modified row from the redo stream and then querying the underlying table to fetch the changed columns. For instance, partial updates on LOBs are supported by identifying the modified row and the LOB column from the redo log, and then querying for the LOB column value for the row from the base table. A similar technique is employed to support UDT.



Note:

Extract only requires fetch for UDT when *not* using native object support.

Such fetch-based support is implemented by issuing a flashback query to the database based on the SCN (System Change Number) at which the transaction committed. The flashback query feature has certain limitations. Certain DDL operations act as barriers such that flashback queries to get data prior to these DDLs do not succeed. Examples of such DDL are `ALTER TABLE MODIFY COLUMN` and `ALTER TABLE DROP COLUMN`.

Thus, in cases where there is Extract capture lag, an intervening DDL may cause fetch requests for data prior to the DDL to fail. In such cases, Extract falls back and fetches the current snapshot of the data for the modified column. There are several limitations to this approach: First, the DDL could have modified the column that Extract needs to fetch (for example, suppose the intervening DDL added a new attribute to the UDT that is being captured). Second, the DDL could have modified one of the columns that Extract uses as a logical row identifier. Third, the table could have been renamed before Extract had a chance to fetch the data.

To prevent fetch-related inconsistencies such as these, take the following precautions while modifying columns.

1. Pause all DML to the table.
2. Wait for Extract to finish capturing all remaining redo, and wait for Replicat to finish processing the captured data from trail. To determine whether Replicat is finished, issue the following command until you see a message that there is no more data to process.

```
INFO REPLICAT group
```

3. Execute the DDL on the source.
4. Resume source DML operations.

Comments in SQL

If a source DDL statement contains a comment in the middle of an object name, that comment will appear at the end of the object name in the target DDL statement. For example:

Source:

```
CREATE TABLE hr./*comment*/emp ...
```

Target:

```
CREATE TABLE hr.emp /*comment*/ ...
```

This does not affect the integrity of DDL synchronization. Comments in any other area of a DDL statement remain in place when replicated.

Compilation Errors

If a `CREATE` operation on a trigger, procedure, function, or package results in compilation errors, Oracle GoldenGate executes the DDL operation on the target anyway. Technically, the DDL operations themselves completed successfully and should be propagated to allow dependencies to be executed on the target. For example in recursive procedures.

Interval Partitioning

DDL replication is unaffected by interval partitioning, because the DDL is implicit. However, this is system generated name so Replicat cannot convert this to the target. I believe this is expected behavior. You must drop the partition on the source. For example:

```
ALTER TABLE employees DROP PARTITION FOR (20);
```

DML or DDL Performed Inside a DDL Trigger

DML or DDL operations performed from within a DDL trigger are not captured.

LogMiner Data Dictionary Maintenance

Oracle recommends that you gather dictionary statistics *after* the Extract is registered (logminer session) and the logminer dictionary is loaded, or after any significant DDL activity on the database.

Guidelines for Configuring DDL Replication for Oracle

Here are the guidelines for configuring Oracle GoldenGate processes to support DDL replication.

Database Privileges

See [Grant User Privileges for Oracle Database 21c and Lower](#).

Parallel Processing

If using parallel Extract and/or Replicat processes, keep related DDL and DML together in the same process stream to ensure data integrity. Configure the processes so that:

- all DDL and DML for any given object are processed by the same Extract group and by the same Replicat group.
- all objects that are relational to one another are processed by the same process group.

For example, if `repe` processes DML for `EMPLOYEES`, then it should also process the DDL for `EMPLOYEES`. If `APPRAISAL` has a foreign key to `EMPLOYEES`, then its DML and DDL operations also should be processed by `repe`.

If an Extract group writes to multiple trails that are read by different Replicat groups, Extract sends all of the DDL to all of the trails. Use each Replicat group to filter the DDL by using the filter options of the `DDL` parameter in the Replicat parameter file.

Object Names

Oracle GoldenGate preserves the database-defined object name, case, and character set. This support preserves single-byte and multibyte names, symbols, and accent characters at all levels of the database hierarchy.

Object names must be fully qualified with their two-part or three-part names when supplied as input to any parameters that support DDL synchronization. You can use the question mark (?) and asterisk (*) wildcards to specify object names in configuration parameters that support DDL synchronization, but the wildcard specification also must be fully qualified as a two-part or three-part name. To process wildcards correctly, the `WILDCARDRESOLVE` parameter is set to `DYNAMIC` by default. If `WILDCARDRESOLVE` is set to anything else, the Oracle GoldenGate process that is processing DDL operations will abend and write the error to the process report.

Data Definitions

Because DDL support requires a like-to-like configuration, the `ASSUMETARGETDEFS` parameter must be used in the Replicat parameter file. Replicat will abend if objects are configured for DDL support and the `SOURCEDEFS` parameter is being used.

For more information, see `ASSUMETARGETDEFS`.

Truncates

`TRUNCATE` statements can be supported as follows:

- As part of the Oracle GoldenGate full DDL support, which supports `TRUNCATE TABLE`, `ALTER TABLE TRUNCATE PARTITION`, and other DDL. This is controlled by the DDL parameter (see [Enabling DDL Support](#).)
- As standalone `TRUNCATE` support. This support enables you to replicate `TRUNCATE TABLE`, but no other DDL. The `GETTRUNCATES` parameter controls the standalone `TRUNCATE` feature. For more information, see *Parameters and Functions Reference for Oracle GoldenGate*.

To avoid errors from duplicate operations, only one of these features can be active at the same time.

Initial Synchronization

To configure DDL replication, start with a target database that is synchronized with the source database. DDL support is compatible with the Replicat initial load method.

Before executing an initial load, disable DDL extraction and replication. DDL processing is controlled by the DDL parameter in the Extract and Replicat parameter files.

After initial synchronization of the source and target data, use all of the source sequence values at least once with `NEXTVAL` before you run the source applications. You can use a script that selects `NEXTVAL` from every sequence in the system. This must be done while Extract is running.

Data Continuity After CREATE or RENAME

To replicate DML operations on new Oracle tables resulting from a `CREATE` or `RENAME` operation, the names of the new tables must be specified in `TABLE` and `MAP` statements in the parameter files. You can use wildcards to make certain that they are included.

To create a new user with `CREATE USER` and then move new or renamed tables into that schema, the new user name must be specified in `TABLE` and `MAP` statements. To create a user `HR` and move new or renamed tables into that schema, the parameter statements could look as follows, depending on whether you want the `HR` objects mapped to the same, or different, schema on the target:

Extract:

```
TABLE HR.*;
```

Replicat:

```
MAP HR.*, TARGET different_schema.*;
```

Understanding DDL Scopes

Database objects are classified into scopes. A scope is a category that defines how DDL operations on an object are handled by Oracle GoldenGate.

The scopes are:

- MAPPED
- UNMAPPED
- OTHER

The use of scopes enables granular control over the filtering of DDL operations, string substitutions, and error handling.

Mapped Scope

Objects that are specified in `TABLE` and `MAP` statements are of `MAPPED` scope. Extraction and replication instructions in those statements apply to both data (DML) and DDL on the specified objects, unless override rules are applied.

For objects in `TABLE` and `MAP` statements, the DDL operations listed in the following table are supported.

Operations	On any of these Objects ¹
CREATE	TABLE ³
ALTER	INDEX
DROP	TRIGGER
RENAME	SEQUENCE
COMMENT ON ²	MATERIALIZED VIEW
	VIEW
	FUNCTION
	PACKAGE
	PROCEDURE
	SYNONYM
	PUBLIC SYNONYM ⁴

Operations	On any of these Objects ¹
GRANT	TABLE
REVOKE	SEQUENCE
	MATERIALIZED VIEW
ANALYZE	TABLE
	INDEX
	CLUSTER

¹ TABLE and MAP do not support some special characters that could be used in an object name affected by these operations. Objects with non-supported special characters are supported by the scopes of UNMAPPED and OTHER.

² Applies to COMMENT ON TABLE, COMMENT ON COLUMN

³ Includes AS SELECT

⁴ Table name must be qualified with schema name.

For Extract, MAPPED scope marks an object for DDL capture according to the instructions in the TABLE statement. For Replicat, MAPPED scope marks DDL for replication and maps it to the object specified by the schema and name in the TARGET clause of the MAP statement. To perform this mapping, Replicat issues ALTER SESSION to set the schema of the Replicat session to the schema that is specified in the TARGET clause. If the DDL contains unqualified objects, the schema that is assigned on the target depends on circumstances described in [Understanding DDL Scopes](#).

Assume the following TABLE and MAP statements:

Extract (source)

```
TABLE hr.employees;  
TABLE hr.emp*;
```

Replicat (target)

```
MAP hr.employees, TARGET hr2.employees2;  
MAP hr.emp*, TARGET hrEMPLOYEES.bak_*;
```

Also assume a source DDL statement of:

```
ALTER TABLE hr.employees ADD notes varchar2(100);
```

In this example, because the source table `fin.expen` is in a MAP statement with a TARGET clause that maps to a different schema and table name, the target DDL statement becomes:

```
ALTER TABLE hr2.employees2 ADD notes varchar2(100);
```

Likewise, the following source and target DDL statements are possible for the second set of TABLE and MAP statements in the example:

Source:

```
CREATE TABLE hr.tabPayables ... ;
```

Target:

```
CREATE TABLE hrBackup.bak_tabPayables ...;
```

When objects are of `MAPPED` scope, you can omit their names from the DDL configuration parameters, unless you want to refine their DDL support further. If you ever need to change the object names in `TABLE` and `MAP` statements, the changes will apply automatically to the DDL on those objects.

If you include an object in a `TABLE` statement, but not in a `MAP` statement, the DDL for that object is `MAPPED` in scope on the source but `UNMAPPED` in scope on the target.

Unmapped Scope

If a DDL operation is supported for use in a `TABLE` or `MAP` statement, but its base object name is not included in one of those parameters, it is of `UNMAPPED` scope.

An object name can be of `UNMAPPED` scope on the source (not in an Extract `TABLE` statement), but of `MAPPED` scope on the target (in a Replicat `MAP` statement), or the other way around. When Oracle DDL is of `UNMAPPED` scope in the Replicat configuration, Replicat will by default do the following:

1. Set the current schema of the Replicat session to the schema of the source DDL object.
2. Execute the DDL as that schema.
3. Restore Replicat as the current schema of the Replicat session.

Other Scope

DDL operations that cannot be mapped are of `OTHER` scope. When DDL is of `OTHER` scope in the Replicat configuration, it is applied to the target with the same schema and object name as in the source DDL.

An example of `OTHER` scope is a DDL operation that makes a system-specific reference, such as DDL that operates on data file names.

Some other examples of `OTHER` scope:

```
CREATE USER joe IDENTIFIED by joe;  
CREATE ROLE ggs_gguser_role IDENTIFIED GLOBALLY;  
ALTER TABLESPACE gg_user TABLESPACE GROUP gg_grp_user;
```

Correctly Identifying Unqualified Object Names in DDL

Extract captures the current schema (also called session schema) that is in effect when a DDL operation is executed. The current container is also captured if the source is a multitenant container database.

The container and schema are used to resolve unqualified object names in the DDL.

Consider the following example:

```
CONNECT ggadmin/PASSWORD  
CREATE TABLE EMPLOYEES (X NUMBER);  
CREATE TABLE EAST.FINANCE(X NUMBER) AS SELECT * FROM EMPLOYEES;
```

In both of those DDL statements, the unqualified table `TAB1` is resolved as `SCOTT.TAB1` based on the current schema `SCOTT` that is in effect during the DDL execution.

There is another way of setting the current schema, which is to set the `current_schema` for the session, as in the following example:

```
CONNECT ggadmin/PASSWORD
ALTER SESSION SET CURRENT_SCHEMA=SRC;
CREATE TABLE EMPLOYEES (X NUMBER);
CREATE TABLE HR.FINANCE(X NUMBER) AS SELECT * FROM EMPLOYEES;
```

In both of those DDL statements, the unqualified table `EMPLOYEES` is resolved as `HR.EMPLOYEES` based on the current schema `HR` that is in effect during the DDL execution.

Extract captures the current schema that is in effect during DDL execution, and it resolves the unqualified object names (if any) by using the current schema. As a result, `MAP` statements specified for Replicat, work correctly for DDL with unqualified object names.

You can also map a source session schema to a different target session schema, if that is required for the DDL to succeed on the target. This mapping is global and overrides any other mappings that involve the same schema names. To map session schemas, use the `DDLOPTIONS` parameter with the `MAPSESSIONSCHEMA` option.

If the default or mapped session schema mapping fails, you can handle the error with the following `DDLERROR` parameter statement, where error 1435 means that the schema does not exist.

```
DDLERROR 1435 IGNORE INCLUDE OPTYPE ALTER OBJTYPE SESSION
```

Enabling DDL Support

Data Definition Language (DDL) is useful in dynamic environments which change constantly.

By default, the status of DDL replication support is as follows:

- On the source, Oracle GoldenGate DDL support is disabled by default. You must configure Extract to capture DDL by using the `DDL` parameter.
- On the target, DDL support is enabled by default, to maintain the integrity of transactional data that is replicated. By default, Replicat will process all DDL operations that the trail contains. If needed, you can use the `DDL` parameter to configure Replicat to ignore or filter DDL operations.

Filtering DDL Replication

By default, all DDL is passed to Extract.

You can use the filtering with DDL parameter method to filter DDL operations so that specific (or all) DDL is applied to the target database according to your requirements. Valid for native DDL capture. This is the preferred method of filtering and is performed within Oracle GoldenGate, and both Extract and Replicat can execute filter criteria. Extract can perform filtering, or it can send the entire DDL to a trail, and then Replicat can perform the filtering. Alternatively, you can filter in a combination of different locations. The `DDL` parameter gives you control over where the filtering is performed, and it also offers more filtering options, including the ability to filter collectively based on the DDL scope (for example, include all `MAPPED` scope).

**Note:**

If a DDL operation fails in the middle of a `TRANSACTION`, it forces a commit, which means that the transaction spanning the DDL is split into two. The first half is committed and the second half can be restarted. If a recovery occurs, the second half of the transaction cannot be filtered since the information contained in the header of the transaction is no longer there.

Filtering with the DDL Parameter

The `DDL` parameter is the main Oracle GoldenGate parameter for filtering DDL within the Extract and Replicat processes.

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, along with other options, to filter the DDL to the required level.

- `DDL` filtering options are valid for a primary Extract that captures from the transaction source.
- 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 DDL filtering criteria, it is recommended that you test your configuration in a test environment before using it in production.

See `DDL` parameter syntax and additional usage guidelines in the *Parameters and Functions Reference for Oracle GoldenGate*.

**Note:**

Before you configure DDL support, it might help to review [How DDL is Evaluated for Processing](#).

Special Filter Cases

This topic describes the special cases that you must consider before creating your DDL filters.

The following are the special cases for creating filter conditions.

DDL EXCLUDE ALL

`DDL EXCLUDE ALL` is a special processing option that is intended primarily for Extract. `DDL EXCLUDE ALL` blocks the replication of DDL operations, but ensures that Oracle GoldenGate continues to keep the object metadata current. When Extract receives DDL directly from the logging server (triggerless DDL capture mode), current metadata is always maintained.

You can use `DDL EXCLUDE ALL` when using a method other than Oracle GoldenGate to apply DDL to the target and 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.

To prevent all DDL metadata and operations from being replicated, omit the `DDL` parameter entirely.

Implicit DDL

User-generated DDL operations can generate implicit DDL operations. For example, the following statement generates two distinct DDL operations.

```
CREATE TABLE customers (custID number, name varchar2(50), address
varchar2(75), address2 varchar2(75), city varchar2(50), state (varchar2(2),
zip number, contact varchar2(50), areacode number(3), phone number(7),
primary key (custID));
```

The first (explicit) DDL operation is the `CREATE TABLE` statement itself.

The second DDL operation is an implicit `CREATE UNIQUE INDEX` statement that creates the index for the primary key. This operation is generated by the database engine, not a user application.

How Oracle GoldenGate Handles Derived Object Names

DDL operations can contain a *base object* name and also a *derived object* name.

A base object is an object that contains data. A derived object is an object that inherits some attributes of the base object to perform a function related to that object. DDL statements that have both base and derived objects are:

- `RENAME` and `ALTER RENAME`
- `CREATE` and `DROP` on an index, synonym, or trigger

Consider the following DDL statement:

```
CREATE INDEX hr.indexPayrollDate ON TABLE hr.tabPayroll (payDate);
```

In this case, the table is the base object. Its name (`hr.tabPayroll`) is the *base name* and is subject to mapping with `TABLE` or `MAP` under the `MAPPED` scope. The derived object is the index, and its name (`hr.indexPayrollDate`) is the *derived name*.

You can map a derived name in its own `TABLE` or `MAP` statement, separately from that of the base object. Or, you can use one `MAP` statement to handle both. In the case of `MAP`, the conversion of derived object names on the target works as follows:

MAP Exists for Base and Derived Objects

If there is a `MAP` statement for the base object and also one for the derived object, the result is an explicit mapping. Assuming the DDL statement includes `MAPPED`, Replicat converts the schema and name of each object according to its own `TARGET` clause. For example, assume the following:

Extract (source)

```
TABLE hr.tab*; TABLE hr.index*;
```

Replicat (target)

```
MAP hr.tab*, TARGET hrBackup.*; MAP hr.index*, TARGET hrIndex.*;
```

Assume the following source DDL statement:

```
CREATE INDEX hr.indexPayrollDate ON TABLE hr.tabPayroll (payDate);
```

The `CREATE INDEX` statement is executed by Replicat on the target as follows:

```
CREATE INDEX hrIndex.indexPayrollDate ON TABLE hrBackup.tabPayroll (payDate);
```

Use an explicit mapping when the index on the target must be owned by a different schema from that of the base object, or when the name on the target must be different from that of the source.

MAP Exists for Derived Object, But Not Base Object

If there is a `MAP` statement for the derived object, but not for the base object, Replicat does not perform any name conversion for either object. The target DDL statement is the same as that of the source. To map a derived object, the choices are:

- Use an explicit `MAP` statement for the base object.
- If names permit, map both base and derived objects in the same `MAP` statement by means of a wildcard.
- Create a `MAP` statement for each object, depending on how you want the names converted.

New Tables as Derived Objects

The following explains how Oracle GoldenGate handles new tables that are created from:

- `RENAME` and `ALTER RENAME`
- `CREATE TABLE AS SELECT`

Prerequisites for Configuring DDL

The `CREATE TABLE AS SELECT` (CTAS) statements include `SELECT` statements and `INSERT` statements that reference any number of underlying objects. By default, Oracle GoldenGate obtains the data for the `AS SELECT` clause from the target database. You can force the CTAS operation to preserve the original inserts using this parameter.



Note:

For this reason, Oracle XMLType tables created from a CTAS (`CREATE TABLE AS SELECT`) statement cannot be supported. For XMLType tables, the row object IDs must match between source and target, which cannot be maintained in this scenario. XMLType tables created by an empty CTAS statement (that does not insert data in the new table) can be maintained correctly.

In addition, you could use the `GETCTASDML` parameter that allows 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.

The objects in the `AS SELECT` clause must exist in the target database, and their names must be identical to the ones on the source.

In a `MAP` statement, Oracle GoldenGate only maps the name of the new table (`CREATE TABLE name`) to the `TARGET` specification, but does not map the names of the underlying objects from the `AS SELECT` clause. There could be dependencies on those objects that could cause data inconsistencies if the names were converted to the `TARGET` specification.

The following shows an example of a `CREATE TABLE AS SELECT` statement on the source and how it would be replicated to the target by Oracle GoldenGate.

```
CREATE TABLE a.tab1 AS SELECT * FROM a.tab2;
```

The `MAP` statement for Replicat is as follows:

```
MAP a.tab*, TARGET a.x*;
```

The target DDL statement that is applied by Replicat is the following:

```
CREATE TABLE a.xtab1 AS SELECT * FROM a.tab2;
```

The name of the table in the `AS SELECT * FROM` clause remains as it was on the source: `tab2` (rather than `xtab2`).

To keep the data in the underlying objects consistent on source and target, you can configure them for data replication by Oracle GoldenGate. In the preceding example, you could use the following statements to accommodate this requirement:

Source

```
TABLE a.tab*;
```

Target

```
MAPEXCLUDE a.tab2  
MAP a.tab*, TARGET a.x*;  
MAP a.tab2, TARGET a.tab2;
```

See [Correctly Identifying Unqualified Object Names in DDL](#).

RENAME and ALTER TABLE RENAME

In `RENAME` and `ALTER TABLE RENAME` operations, the base object is always the new table name. In the following example, the base object name is considered to be `index_paydate`.

```
ALTER TABLE hr.indexPayrollDate RENAME TO index_paydate;
```

or...

```
RENAME hr.indexPayrollDate TO index_paydate;
```

The derived object name is `hr.indexPayrollDate`.

Disabling the Mapping of Derived Objects

Use the `DDLOPTIONS` parameter with the `NOMAPDERIVED` option to prevent the conversion of the name of a derived object according to a `TARGET` clause of a `MAP` statement that includes it. `NOMAPDERIVED` overrides any explicit `MAP` statements that contain the name of the base or derived object. Source DDL that contains derived objects is replicated to the target with the same schema and object names as on the source.

The following table shows the results of `MAPDERIVED` compared to `NOMAPDERIVED`, based on whether there is a `MAP` statement just for the base object, just for the derived object, or for both.

Using DDL String Substitution

This feature provides a convenience for changing and mapping directory names, comments, and other things that are not directly related to data structures. For example, you could substitute one tablespace name for another, or substitute a string within comments. String substitution is controlled by the `DDLSUBST` parameter. For more information, see *Parameters and Functions Reference for Oracle GoldenGate*.

**Note:**

Before you create a `DDL SUBST` parameter statement, it might help to review [How DDL is Evaluated for Processing](#).

Controlling the Propagation of DDL to Support Different Topologies

To support bidirectional and cascading replication configurations, it is important for Extract to be able to identify the DDL that is performed by Oracle GoldenGate and by other applications, such as the local business applications.

Depending on the configuration that you want to deploy, it might be appropriate to capture one or both of these sources of DDL on the local system.

**Note:**

Oracle GoldenGate DDL consists of `ALTER TABLE` statements performed by Extract to create log groups and the DDL that is performed by Replicat to replicate source DDL changes.

The following options of the `DDLOPTIONS` parameter control whether DDL on the local system is captured by Extract and then sent to a remote system, assuming Oracle GoldenGate DDL support is configured and enabled:

- The `GETREPLICATES` and `IGNOREREPLICATES` options control whether Extract captures or ignores the DDL that is generated by Oracle GoldenGate. The default is `IGNOREREPLICATES`, which does not propagate the DDL that is generated by Oracle GoldenGate. To identify the DDL operations that are performed by Oracle GoldenGate, the following comment is part of each Extract and Replicat DDL statement:

```
/* GOLDENGATE_DDL_REPLICATION */
```

- The `GETAPPLOPS` and `IGNOREAPPLOPS` options control whether Extract captures or ignores the DDL that is generated by applications other than Oracle GoldenGate. The default is `GETAPPLOPS`, which propagates the DDL from local applications (other than Oracle GoldenGate).

The result of these default settings is that Extract ignores its own DDL and the DDL that is applied to the local database by a local Replicat, so that the DDL is not sent back to its source, and Extract captures all other DDL that is configured for replication. The following is the default `DDLOPTIONS` configuration.

```
DDLOPTIONS GETAPPLOPS, IGNOREREPLICATES
```

This behavior can be modified. See the following topics:

Propagating DDL in Active-Active (Bidirectional) Configurations

Oracle GoldenGate supports active-active DDL replication between two systems. For an active-active bidirectional replication, the following must be configured in the Oracle GoldenGate processes:

1. DDL that is performed by a business application on one system must be replicated to the other system to maintain synchronization. To satisfy this requirement, include the `GETAPPLOPS` option in the `DDLOPTIONS` statement in the Extract parameter files on both systems.
2. DDL that is applied by Replicat on one system must be captured by the local Extract and sent back to the other system. To satisfy this requirement, use the `GETREPLICATES` option in the `DDLOPTIONS` statement in the Extract parameter files on both systems.

Note:

An internal Oracle GoldenGate token will cause the actual Replicat DDL statement itself to be ignored to prevent loopback. The purpose of propagating Replicat DDL back to the original system is so that the Replicat on that system can update its object metadata cache, in preparation to receive incoming DML, which will have the new metadata.

3. Each Replicat must be configured to update its object metadata cache whenever the remote Extract sends over a captured Replicat DDL statement. To satisfy this requirement, use the `UPDATEMETADATA` option in the `DDLOPTIONS` statement in the Replicat parameter files on both systems.

The resultant `DDLOPTIONS` statements should look as follows:

Extract (primary and secondary)

```
DDLOPTIONS GETREPLICATES, GETAPPLOPS
```

Replicat (primary and secondary)

```
DDLOPTIONS UPDATEMETADATA
```

WARNING:

Before you allow new DDL or DML to be issued for the same object(s) as the original DDL, allow time for the original DDL to be replicated to the remote system and then captured again by the Extract on that system. This will ensure that the operations arrive in correct order to the Replicat on the original system, to prevent DML errors caused by metadata inconsistencies. See the following diagram for more information.

For more information, see *Parameters and Functions Reference for Oracle GoldenGate*.

Prerequisites for Configuring DDL

In a cascading configuration, use the following setting for `DDLOPTIONS` in the Extract parameter file on each intermediary system. This configuration forces Extract to capture the DDL from Replicat on an intermediary system and cascade it to the next system downstream.

```
DDLOPTIONS GETREPLICATES, IGNOREAPPLOPS
```

For more information about `DDLOPTIONS`, see `DDLOPTIONS`.

Add Supplemental Log Groups Automatically

Use the `DDLOPTIONS` parameter with the `ADDTRANDATA` option for performing tasks described in this topic.

You can perform the following tasks using the `DDLOPTIONS`:

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

To use `DDLOPTIONS ADDSCHEMATRANDATA`, the `ADD SCHEMATRANDATA` command must be issued on the Admin Client to enable schema-level supplemental logging.

`DDLOPTIONS ADDTRANDATA` is not supported for multitenant container databases, see [Configuring Logging Properties](#) for more information.

Removing Comments from Replicated DDL

You can use the `DDLOPTIONS` parameter with the `REMOVECOMMENTS BEFORE` and `REMOVECOMMENTS AFTER` options to prevent comments that were used in the source DDL from being included in the target DDL.

By default, comments are not removed, so that they can be used for string substitution.

Replicating an IDENTIFIED BY Password

Use the `DDLOPTIONS` parameter with the `DEFAULTUSERPASSWORDALIAS` and `REPLICATEPASSWORD | NOREPLICATEPASSWORD` options to control how the password of a replicated `{CREATE | ALTER} USER name IDENTIFIED BY password` statement is handled. These options must be used together.

See the `USEPASSWORDVERIFIERLEVEL` option of `DDLOPTIONS` for important information about specifying the password verifier when Replicat operates against an Oracle 10g or 11g database.

**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.

How DDL is Evaluated for Processing

Learn about the order in which different criteria in the Oracle GoldenGate parameters are processed, and the differences between how Extract and Replicat each process the DDL.

Extract

1. Extract captures a DDL statement.
2. Extract separates comments, if any, from the main statement.
3. Extract searches for the `DDL` parameter. (This example assumes it exists.)
4. Extract searches for the `IGNOREREPLICATES` parameter. If it is present, and if Replicat produced this DDL on this system, Extract ignores the DDL statement. (This example assumes no Replicat operations on this system.)
5. Extract determines whether the DDL statement is a `RENAME`. If so, the rename is flagged internally.
6. Extract gets the base object name and, if present, the derived object name.
7. If the statement is a `RENAME`, Extract changes it to `ALTER TABLE RENAME`.
8. Extract searches for the `DDLOPTIONS REMOVECOMMENTS BEFORE` parameter. If it is present, Extract removes the comments from the DDL statement, but stores them in case there is a `DDL INCLUDE` or `DDL EXCLUDE` clause that uses `INSTR` or `INSTRCOMMENTS`.
9. Extract determines the DDL scope: `MAPPED`, `UNMAPPED` or `OTHER`:
 - It is `MAPPED` if the operation and object types are supported for mapping, and the base object name and/or derived object name (if `RENAME`) is in a `TABLE` parameter.
 - It is `UNMAPPED` if the operation and object types are not supported for mapping, and the base object name and/or derived object name (if `RENAME`) is not in a `TABLE` parameter.
 - Otherwise the operation is identified as `OTHER`.
10. Extract checks the `DDL` parameter for `INCLUDE` and `EXCLUDE` clauses, and it evaluates the `DDL` parameter criteria in those clauses. All options must evaluate to `TRUE` in order for the `INCLUDE` or `EXCLUDE` to evaluate to `TRUE`. The following occurs:
 - If an `EXCLUDE` clause evaluates to `TRUE`, Extract discards the DDL statement and evaluates another DDL statement. In this case, the processing steps start over.
 - If an `INCLUDE` clause evaluates to `TRUE`, or if the `DDL` parameter does not have any `INCLUDE` or `EXCLUDE` clauses, Extract includes the DDL statement, and the processing logic continues.
11. Extract searches for a `DDLSUBST` parameter and evaluates the `INCLUDE` and `EXCLUDE` clauses. If the criteria in those clauses add up to `TRUE`, Extract performs string substitution. Extract evaluates the DDL statement against each `DDLSUBST` parameter in the parameter

file. For all true `DDLSUBST` specifications, Extract performs string substitution in the order that the `DDLSUBST` parameters are listed in the file.

12. Now that `DDLSUBT` has been processed, Extract searches for the `REMOVECOMMENTS AFTER` parameter. If it is present, Extract removes the comments from the DDL statement.
13. Extract searches for `DDLOPTIONS ADDTRANDATA`. If it is present, and if the operation is `CREATE TABLE`, Extract issues the `ALTER TABLE name ADD SUPPLEMENTAL LOG GROUP` command on the table.
14. Extract writes the DDL statement to the trail.

Viewing DDL Report Information

By default, Oracle GoldenGate shows basic statistics about DDL at the end of the Extract and Replicat reports.

To enable expanded DDL reporting, use the `DDLOPTIONS` parameter with the `REPORT` option. Expanded reporting includes the following information about DDL processing:

- A step-by-step history of the DDL operations that were processed by Oracle GoldenGate.
- The DDL filtering and processing parameters that are being used.

Expanded DDL report information increases the size of the report file, but it might be useful in certain situations, such as for troubleshooting or to determine when an `ADD TRANDATA` to add supplemental logging was applied.

To view a report, use the `VIEW REPORT` command.

```
VIEW REPORT group
```

Viewing DDL Reporting in Replicat

The Replicat report lists:

- The entire syntax and source Oracle GoldenGate SCN of each DDL operation that Replicat processed from the trail. You can use the source SCN for tracking purposes, especially when there are restores from backup and Replicat is positioned backward in the trail.
- A subsequent entry that shows the scope of the operation (`MAPPED`, `UNMAPPED`, `OTHER`) and how object names were mapped in the target DDL statement, if applicable.
- Another entry that shows how processing criteria was applied.
- Additional entries that show whether the operation succeeded or failed, and whether or not Replicat applied error handling rules.

The following excerpt from a Replicat report illustrates a sequence of steps, including error handling:

```
2023-09-06 18:50:13 INFO OGG-01487 DDL found, operation [create table
hr.employees(a int primary key, b int) (size 45)], start SCN [3344441],
commit SCN [3344461] instance [ (1)], DDL seqno [0], marker seqno [0].
2023-09-06 18:50:13 INFO OGG-10451 DDL operation included [INCLUDE MAPPED],
optype [CREATE], objtype [TABLE], catalog "CDBA_PDB01", objowner "HR",
objname "EMPLOYEES".
2023-09-06 18:50:13 INFO OGG-01487 DDL found, operation [create table
HR.EMPLOYEES_BAK (a int primary key, b int) (size 45)], start SCN [3344467],
```

```
commit SCN [3344486] instance [ (1)], DDL seqno [0], marker seqno [0].
2023-09-06 18:50:13 INFO OGG-10452 DDL operation excluded [EXCLUDE OBJNAME
HR.EMPLOYEES_BAK], optype [CREATE], objtype [EMPLOYEES_BAK], catalog
"CDBA_PDB01", objowner "HR", objname "EMPLOYEES_BAK".
```

Viewing DDL Reporting in Extract

The Extract report lists the following:

- The entire syntax of each captured DDL operation, the start and end SCN, the Oracle instance, the DDL sequence number (from the `SEQNO` column of the history table), and the size of the operation in bytes.
- A subsequent entry that shows how processing criteria was applied to the operation, for example string substitution or `INCLUDE` and `EXCLUDE` filtering.
- Another entry showing whether the operation was written to the trail or excluded.

The following excerpt, taken from an Extract report, shows an included operation and an excluded operation. There is a report message for the included operation, but not for the excluded one.

```
2011-01-20 15:11:41 GGS INFO
2100 DDL found, operation
      [create table hr.employees
      (empId number (10) not null,
      Phone Number number,
      Designation varchar2(100),
      Date date,
      primary key (empId))],
      start SCN [1186754], commit SCN [1186772] instance [test11g (1)], DDL seqno
[4134].
```

```
2011-01-20 15:11:41 GGS INFO      2100 DDL operation included [INCLUDE
OBJNAME employees*], optype [CREATE], objtype [TABLE], objname
[QATEST1.EMPLOYEES].
```

```
2011-01-20 15:11:41 GGS INFO      2100 DDL operation written to extract
trail file.
```

```
2011-01-20 15:11:42 GGS INFO      2100 Successfully added TRANDATA for table
with the key, table [QATEST1.EMPLOYEES], operation [ALTER TABLE
"QATEST1"."EMPLOYEES" ADD SUPPLEMENTAL LOG GROUP "GGS_EMPLOYEES_53475" (MYID)
ALWAYS /* GOLDENGATE_DDL_REPLICATION */ ].
```

```
2011-01-20 15:11:43 GGS INFO      2100 DDL found, operation [create table
EMPLOYEESTemp (
      vid varchar2(100),
      someDate date,
      primary key (vid)) ],
      start SCN [1186777], commit SCN [1186795] instance [test11g (1)], DDL seqno
[4137].
```

```
2011-01-20 15:11:43 GGS INFO      2100 DDL operation excluded [EXCLUDE
OBJNAME EMPLOYEESTemp OPTYPE CREATE], optype [CREATE], objtype [TABLE],
objname [QATEST1.EMPLOYEESTEMP].
```

Statistics in the Process Reports

You can send current statistics for DDL processing to the Extract and Replicat reports by using the `SEND` command in Admin Client.

```
SEND {EXTRACT | REPLICAT} group REPORT
```

The statistics show totals for:

- All DDL operations
- Operations that are `MAPPED` in scope
- Operations that are `UNMAPPED` in scope
- Operations that are `OTHER` in scope
- Operations that were excluded (number of operations minus included ones)
- Errors (Replicat only)
- Retried errors (Replicat only)
- Discarded errors (Replicat only)
- Ignored operations (Replicat only)

Tracing DDL Processing

If you open a support case with Oracle GoldenGate Technical Support, you might be asked to turn on tracing. `TRACE` and `TRACE2` control DDL tracing.

Procedural Replication

Learn about procedural replication and how to configure it.

About Procedural Replication

Procedural replication is available with Oracle database only. Oracle GoldenGate uses procedural replication to replicate Oracle Database supplied PL/SQL procedures avoiding the shipping and applying of high volume records usually generated by these operations. Procedural replication implements dictionary changes that control user and session behavior and the swapping of objects in dictionary.

Procedural replication is not related to the replication of the `CREATE`, `ALTER`, and `DROP` statements (or DDL), rather it is the replication of a procedure call like:

```
CALL procedure_name(arg1, arg2, ...);
```

As opposed to:

```
exec procedure_name(arg1, arg2, ...)
```

After you enable procedural replication, calls to procedures in Oracle Database supplied packages at one database are replicated to one or more other databases and then executed at those databases. For example, a call to subprograms in the `DBMS_REDEFINITION` package can perform an online redefinition of a table. If the table is replicated at several databases, and if

you want the same online redefinition to be performed on the table at each database, then you can make the calls to the subprograms in the `DBMS_REDEFINITION` package at one database, and Oracle GoldenGate can replicate those calls to the other databases.

To support procedural replication, your Oracle Database should be configured to identify procedures that are enabled for this optimization.

To use procedural replication, the following prerequisites must be met:

- Oracle GoldenGate with Extract and Replicat.
- System supplied packages are only working in combination with DML and DDL.

Procedural Replication Process Overview

Procedural replication uses a trail record to ensure that sufficient information is encapsulated with the record.

To use Oracle GoldenGate procedural replication, you need to enable it. Your Oracle Database must have a built in mechanism to identify the procedures that are enabled for this optimization.

PL/SQL pragmas are used to indicate which procedures can be replicated. When the pragma is specified, a callback is made to Logminer on entry and exit from the routine. The callback provides the name of the procedure call and arguments and indicates if the procedure exited successfully or with an error. Logminer augments the redo stream with the information from the callbacks. For supported procedures, the normal redo generated by the procedure is suppressed, and only the procedure call is replicated.

A new trail record is generated to identify procedural replication. This trail record leverages existing trail column data format for arguments passed to PL/SQL procedures. For LOBs, data is passed in chunks similar to existing trail format for LOBs. This trail record has sufficient information to replay the procedure as-is on the target.

When you enable procedural replication, it prevents writing of individual records impacted by the procedure to the trail file.

If an error is encountered when applying a PL/SQL procedure, Replicat can replay the entire PL/SQL procedure.

Determining Whether Procedural Replication Is On

Use the `GG_PROCEDURE_REPLICATION_ON` function in the `DBMS_GOLDENGATE_ADM` package to determine whether Oracle GoldenGate procedural replication is on or off.

If you want to use Oracle GoldenGate in an Oracle Database Vault environment with procedural replication, then you must set the appropriate privileges. See *Oracle Database Vault Administrator's Guide*.

To enable procedural replication:

1. Connect to the database as `sys` (`sqlplus`, `sqlcl`, `sqldeveloper`) not as an Oracle GoldenGate administrator.
2. Run the `GG_PROCEDURE_REPLICATION_ON` function.

Example 9-1 Running the `GG_PROCEDURE_REPLICATION_ON` Function

```
SET SERVEROUTPUT ON
DECLARE
  on_or_off    NUMBER;
```

```
BEGIN
  on_or_off := DBMS_GOLDENGATE_ADM.GG_PROCEDURE_REPLICATION_ON;
  IF on_or_off=1 THEN
    DBMS_OUTPUT.PUT_LINE('Oracle GoldenGate procedural replication is ON.');
```

ELSE

```
    DBMS_OUTPUT.PUT_LINE('Oracle GoldenGate procedural replication is OFF.');
```

END IF;

```
END;
/
```

Enabling and Disabling Supplemental Logging

Oracle GoldenGate provides commands to allow you to enable or disable procedural supplemental logging.

To enable supplemental logging:

1. Connect to the source database as the Oracle GoldenGate administrator with `DBLOGIN`.

```
CONNECT https://localhost:9000 DEPLOYMENT demo AS ggadmin PASSWORD adminpw
```

```
DBLOGIN USERIDALIAS ggeast DOMAIN OracleGoldenGate
```

2. Add supplemental logging for procedural replication.

```
ADD PROCEDURETRANDATA
```

The output shows:

```
INFO OGG-13005 PROCEDURETRANDATA supplemental logging has been enabled.
```

Supplemental logging is enabled for procedure replication.

To disable supplemental logging:

1. Connect to the source database as the Oracle GoldenGate administrator with `dblogin`.

```
CONNECT https://localhost:9000 DEPLOYMENT demo AS ggadmin PASSWORD adminpw
```

```
DBLOGIN USERIDALIAS ggeast DOMAIN OracleGoldenGate
```

2. Remove supplemental logging for procedure replication.

```
DELETE PROCEDURETRANDATA
```

Supplemental logging is disabled for procedure replication.

To view information about supplemental logging:

1. Connect to the source database as the Oracle GoldenGate administrator with dblogin.

```
CONNECT https://localhost:9000 DEPLOYMENT demo AS ggadmin PASSWORD adminpw
```

```
DBLOGIN USERIDALIAS ggeast DOMAIN OracleGoldenGate
```

2. Display supplemental logging information for procedure replication.

```
INFO PROCEDURETRANDATA
```

Supplemental logging information for procedure replication is displayed.

Filtering Features for Procedural Replication

You can specify which procedures and packages you want to include or exclude for procedure replication.

You group supported packages and procedures using feature groups. You use the procedure parameter with the `INCLUDE` or `EXCLUDE` keyword to filter features for procedure replication.

In the procedure parameter, `INCLUDE` or `EXCLUDE` specify the beginning of a filtering clause. They specify the procedures to replicate (`INCLUDE`) or filter out (`EXCLUDE`). The filtering clause must consist of the `INCLUDE ALL_SUPPORTED` or `EXCLUDE ALL_SUPPORTED` keyword followed by any valid combination of the other filtering options of the procedure parameter. The `EXCLUDE` filter takes precedence over any `INCLUDE` filters that contain the same criteria.



Note:

When replicating Oracle Streams Advanced Queuing (AQ) procedures, you must use the `RULE` option in your parameter file as follows:

```
PROCEDURE INCLUDE FEATURE ALL_SUPPORTED
```

or

```
PROCEDURE INCLUDE FEATURE AQ, RULE
```

Do not use `PROCEDURE INCLUDE FEATURE AQ` without the `RULE` option.

Including all system supplied packages at Extract:

1. Connect to Extract in the source database.

```
EXTRACT edba
```

```
USERIDALIAS admin_dbA DOMAIN ORADEV
```

2. Create a new trail file.

```
EXTTRAIL ea
```

3. Enable procedure replication, if not already done.

```
TRANLOGOPTIONS INTEGRATEDPARAMS (ENABLE_PROCEDURAL_REPLICATION Y)
```

4. Include filter for procedure replication.

```
PROCEDURE INCLUDE FEATURE ALL_SUPPORTED
```

You have successfully included all system supplied packages for procedure replication.

Excluding specific packages at Replicat:

1. Connect to Replicat in the target database.

```
REPLICAT rdba  
USERIDALIAS admin_dbBDOMAIN ORADEV
```

2. Include filter for procedure replication.

```
PROCEDURE EXCLUDE FEATURE RLS
```

You have successfully excluded specific packages for procedure replication.

Handling Procedural Replication Errors

Procedural replication uses `REPERROR` parameter to configure the behavior of Replicat when an procedural error occurs.

By default, Replicat will abend when a procedural replication occurs so using the following steps sets up error handling:

1. Connect to Replicat in the target database.

```
REPLICAT rdba  
USERIDALIAS admin_dbBDOMAIN ORADEV
```

2. Include filter for procedure replication.

```
PROCEDURE EXCLUDE FEATURE RLS
```

3. Specify error handling parameter, see `REPERROR` in *Parameters and Functions Reference for Oracle GoldenGate* for other options.

```
REPERROR (PROCEDURE, DISCARD)
```

You have successfully handled errors for procedural replication.

Listing the Procedures Supported for Oracle GoldenGate Procedural Replication

The `DBA_GG_SUPPORTED_PROCEDURES` view displays information about the supported packages for Oracle GoldenGate procedural replication.

When a procedure is supported and Oracle GoldenGate procedural replication is on, calls to the procedure are replicated, unless the procedure is excluded specifically.

1. Connect to the database as `sys` (`sqlplus`, `sqlcl`, `sqldeveloper`) not as an Oracle GoldenGate administrator.
2. Query the `DBA_GG_SUPPORTED_PROCEDURES` view.

Example 9-2 Displaying Information About the Packages Supported for Oracle GoldenGate Procedural Replication

This query displays the following information about the packages:

- The owner of each package
- The name of each package
- The name of each procedure
- The minimum database release from which the procedure is supported

- Whether there is an exclusion rule that prevents the procedure from being replicated for some database objects

```

COLUMN OWNER FORMAT A10
COLUMN PACKAGE_NAME FORMAT A15
COLUMN PROCEDURE_NAME FORMAT A15
COLUMN MIN_DB_VERSION FORMAT A14
COLUMN EXCLUSION_RULE_EXISTS FORMAT A14

```

```

SELECT OWNER,
       PACKAGE_NAME,
       PROCEDURE_NAME,
       MIN_DB_VERSION,
       EXCLUSION_RULE_EXISTS
FROM DBA_GG_SUPPORTED_PROCEDURES;

```

Your output looks similar to the following:

OWNER	PACKAGE_NAME	PROCEDURE_NAME	MIN_DB_VERSION	EXCLUSION_RULE
XDB	DBMS_XDB_CONFIG	ADDTRUSTMAPPING	12.2	NO
CTXSYS	CTX_DDL	ALTER_INDEX	12.2	NO
SYS	DBMS_FGA	DROP_POLICY	12.2	NO
SYS	XS_ACL	DELETE_ACL	12.2	NO
.				
.				
.				

Monitoring Oracle GoldenGate Procedural Replication

A set of data dictionary views enable you to monitor Oracle GoldenGate procedural replication.

You can use the following views to monitor Oracle GoldenGate procedural replication:

View	Description
DBA_GG_SUPPORTED_PACKAGES	Provides details about supported packages for Oracle GoldenGate procedural replication. When a package is supported and Oracle GoldenGate procedural replication is on, calls to subprograms in the package are replicated.
DBA_GG_SUPPORTED_PROCEDURES	Provides details about the procedures that are supported for Oracle GoldenGate procedural replication.
DBA_GG_PROC_OBJECT_EXCLUSION	Provides details about all database objects that are on the exclusion list for Oracle GoldenGate procedural replication. A database object is added to the exclusion list using the <code>INSERT_PROCREP_EXCLUSION_OBJ</code> procedure in the <code>DBMS_GOLDENGATE_ADM</code> package. When a database object is on the exclusion list, execution of a subprogram in the package is not replicated if the subprogram operates on the excluded object.

1. Connect to the database as `sys` (`sqlplus`, `sqlcl`, or `sqldeveloper`) not as an Oracle GoldenGate administrator.
2. Query the views related to Oracle GoldenGate procedural replication.

Execute Commands, Stored Procedures, and Queries with SQLEXEC

The `SQLEXEC` parameter of Oracle GoldenGate enables Extract and Replicat to communicate with the database to do the following:

- Execute a database command, stored procedure, or SQL query to perform a database function, return results (`SELECT` statements) or perform DML (`INSERT`, `UPDATE`, `DELETE`) operations.
- Retrieve output parameters from a procedure for input to a `FILTER` or `COLMAP` clause.



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.

Performing Processing with SQLEXEC

`SQLEXEC` extends the functionality of both Oracle GoldenGate and the database by allowing Oracle GoldenGate to use the native SQL of the database to execute custom processing instructions.

- Stored procedures and queries can be used to select or insert data into the database, to aggregate data, to denormalize or normalize data, or to perform any other function that requires database operations as input. Oracle GoldenGate supports stored procedures that accept input and those that produce output.
- Database commands can be issued to perform database functions required to facilitate Oracle GoldenGate processing, such as disabling triggers on target tables and then enabling them again.

Using SQLEXEC

The `SQLEXEC` parameter can be used as follows:

- as a clause of a `TABLE` or `MAP` statement
- as a standalone parameter at the root level of the Extract or Replicat parameter file.

Apply SQLEXEC as a Standalone Statement

When used as a standalone parameter statement in the Extract or Replicat parameter file, `SQLEXEC` can execute a stored procedure, query, or database command. As such, it need not be tied to any specific table and can be used to perform general SQL operations.

For example, if the Oracle GoldenGate database user account is configured to time-out when idle, you could use `SQLEXEC` to execute a query at a defined interval, so that Oracle GoldenGate does not appear idle. As another example, you could use `SQLEXEC` to issue an

essential database command, such as to disable target triggers. A standalone `SQLEXEC` statement cannot accept input parameters or return output parameters.

Parameter syntax	Purpose
<code>SQLEXEC 'call procedure_name()'</code>	Execute a stored procedure
<code>SQLEXEC 'sql_query'</code>	Execute a query
<code>SQLEXEC 'database_command'</code>	Execute a database command

Argument	Description
<code>'call procedure_name ()'</code>	<p>Specifies the name of a stored procedure to execute. The statement must be enclosed within single quotes.</p> <p>Example:</p> <pre>SQLEXEC 'call prc_job_count ()'</pre>
<code>'sql_query'</code>	<p>Specifies the name of a query to execute. The query must be contained all on one line and enclosed within single quotes.</p> <p>Specify case-sensitive object names the way they are stored in the database, such as within double quotes for Oracle object names that are case-sensitive.</p> <pre>SQLEXEC 'SELECT "col1" from "schema"."table"'</pre>
<code>'database_command'</code>	<p>Specifies a database command to execute. Must be a valid command for the database.</p>

`SQLEXEC` provides options to control processing behavior, memory usage, and error handling. For more information, see `SQLEXEC` in the *Parameters and Functions Reference for Oracle GoldenGate*.

Apply `SQLEXEC` within a `TABLE` or `MAP` Statement

When used within a `TABLE` or `MAP` statement, `SQLEXEC` can pass and accept parameters. It can be used for procedures and queries, but not for database commands.

Syntax

This syntax executes a procedure within a `TABLE` or `MAP` statement.

```
SQLEXEC (SPNAME sp_name,  
[ID logical_name,]  
{PARAMS param_spec | NOPARAMS})
```

Argument	Description
SPNAME	Required keyword that begins a clause to execute a stored procedure.
<i>sp_name</i>	Specifies the name of the stored procedure to execute.
ID <i>logical_name</i>	Defines a logical name for the procedure. Use this option to execute the procedure multiple times within a <code>TABLE</code> or <code>MAP</code> statement. Not required when executing a procedure only once.
PARAMS <i>param_spec</i> NOPARAMS	Specifies whether or not the procedure accepts parameters. One of these options must be used (see Using Input and Output Parameters).

Syntax

This syntax executes a query within a `TABLE` or `MAP` statement.

```
SQLEXEC (ID logical_name, QUERY ' query ',  
{PARAMS param_spec | NOPARAMS})
```

Argument	Description
ID <i>logical_name</i>	Defines a logical name for the query. A logical name is required in order to extract values from the query results. ID <i>logical_name</i> references the column values returned by the query.
QUERY ' <i>sql_query</i> '	Specifies the SQL query syntax to execute against the database. It can either return results with a <code>SELECT</code> statement or change the database with an <code>INSERT</code> , <code>UPDATE</code> , or <code>DELETE</code> statement. The query must be within single quotes and must be contained all on one line. Specify case-sensitive object names the way they are stored in the database, such as within quotes for Oracle case-sensitive names. SQLEXEC 'SELECT "col1" from "schema"."table"'
PARAMS <i>param_spec</i> NOPARAMS	Defines whether or not the query accepts parameters. One of these options must be used (see Using Input and Output Parameters).

If you want to execute a query on a table residing on a different database than the current database, then the different database name has to be specified with the table. The delimiter between the database name and the tablename should be a colon (:).

The following are some example use cases:

```
select col1 from db1:tab1  
select col2 from db2:schema2.tab2
```

```
select col3 from tab3
select col3 from schema4.tab4
```

Using Input and Output Parameters

Oracle GoldenGate provides options for passing input and output values to and from a procedure or query that is executed with `SQLEXEC` within a `TABLE` or `MAP` statement.

Passing Values to Input Parameters

To pass data values to input parameters within a stored procedure or query, use the `PARAMS` option of `SQLEXEC`.

Syntax

```
PARAMS ([OPTIONAL | REQUIRED] param = {source_column | function}
[, ...] )
```

Where:

- `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.
- `REQUIRED` indicates that a parameter value must be present. If the parameter value is not present, the SQL will not be executed.
- `param` is one of the following:
 - For a stored procedure, it is the name of any parameter in the procedure that can accept input, such as a column in a lookup table.
 - For an Oracle query, it is the name of any input parameter in the query excluding the leading colon. For example, `:param1` would be specified as `param1` in the `PARAMS` clause.
 - For a non-Oracle query, it is `pn`, where `n` is the number of the parameter within the statement, starting from 1. For example, in a query with two parameters, the `param` entries are `p1` and `p2`.
- `{source_column | function}` is the column or Oracle GoldenGate conversion function that provides input to the procedure.

Passing Values to Output Parameters

To pass values from a stored procedure or query as input to a `FILTER` or `COLMAP` clause, use the following syntax:

Syntax

```
{procedure_name | logical_name}.parameter
```

Where:

- `procedure_name` is the actual name of the stored procedure. Use this argument only if executing a procedure one time during the life of the current Oracle GoldenGate process.
- `logical_name` is the logical name specified with the `ID` option of `SQLEXEC`. Use this argument if executing a query or a stored procedure that will be executed multiple times.

- *parameter* is either the name of the parameter or `RETURN_VALUE`, if extracting returned values.

SQLEXEC Examples Using Parameters

These examples use stored procedures and queries with input and output parameters.



Note:

Additional `SQLEXEC` options are available for use when a procedure or query includes parameters. See `SQLEXEC` in the *Parameters and Functions Reference for Oracle GoldenGate*.

Example 9-3 SQLEXEC with a Stored Procedure

This example uses `SQLEXEC` to run a stored procedure named `LOOKUP` that performs a query to return a description based on a code. It then maps the results to a target column named `NEWACCT_VAL`.

```
CREATE OR REPLACE PROCEDURE LOOKUP
(CODE_PARAM IN VARCHAR2, DESC_PARAM OUT VARCHAR2)
BEGIN
    SELECT DESC_COL
    INTO DESC_PARAM
    FROM LOOKUP_TABLE
    WHERE CODE_COL = CODE_PARAM
END;
```

Contents of `MAP` statement:

```
MAP sales.account, TARGET sales.newacct, &
  SQLEXEC (SPNAME lookup, PARAMS (code_param = account_code)), &
  COLMAP (newacct_id = account_id, newacct_val = lookup.desc_param);
```

`SQLEXEC` executes the `LOOKUP` stored procedure. Within the `SQLEXEC` clause, the `PARAMS (code_param = account_code)` statement identifies `code_param` as the procedure parameter to accept input from the `account_code` column in the `account` table.

Replicat executes the `LOOKUP` stored procedure prior to executing the column map, so that the `COLMAP` clause can extract and map the results to the `newacct_val` column.

Example 9-4 SQLEXEC with a Query

This example implements the same logic as used in the previous example, but it executes a SQL query instead of a stored procedure and uses the `@GETVAL` function in the column map.

A query must be on one line. To split an Oracle GoldenGate parameter statement into multiple lines, an ampersand (&) line terminator is required.

Query for an Oracle database:

```
MAP sales.account, TARGET sales.newacct, &
  SQLEXEC (ID lookup, &
```

```

QUERY 'select desc_col 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));

```

Query for a non-Oracle database:

```

MAP sales.account, TARGET sales.newacct, &
SQLEXEC (ID lookup, &
QUERY 'select desc_col desc_param from lookup_table where code_col = ?', &
PARAMS (p1 = account_code)), &
COLMAP (newacct_id = account_id, newacct_val = &
@getval (lookup.desc_param));

```

Handling SQLEXEC Errors

There are two types of error conditions to consider when implementing `SQLEXEC`:

- The column map requires a column that is missing from the source database operation. This can occur for an update operation if the database only logs the values of columns that changed, rather than all of the column values. By default, when a required column is missing, or when an Oracle GoldenGate column-conversion function results in a "column missing" condition, the stored procedure does not execute. Subsequent attempts to extract an output parameter from the stored procedure results in a "column missing condition" in the `COLMAP` or `FILTER` clause.
- The database generates an error.

Handling Database Errors

Use the `ERROR` option in the `SQLEXEC` clause to direct Oracle GoldenGate to respond in one of the following ways:

Table 9-1 ERROR Options

Action	Description
IGNORE	Causes Oracle GoldenGate to ignore all errors associated with the stored procedure or query and continue processing. Any resulting parameter extraction results in a "column missing" condition. 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 <code>REPERROR</code> parameter specified in the Replicat parameter file. Oracle GoldenGate continues processing other stored procedures or queries associated with the current <code>TABLE</code> or <code>MAP</code> statement before processing the error.
FINAL	Performs in a similar way to <code>RAISE</code> except that when an error associated with a procedure or query is encountered, any remaining stored procedures and queries are bypassed. Error processing is called immediately after the error.
FATAL	Causes Oracle GoldenGate to abend immediately upon encountering an error associated with a procedure or query.

Handling Missing Column Values

Use the `@COLTEST` function to test the results of the parameter that was passed, and then map an alternative value for the column to compensate for missing values, if desired. Otherwise, to ensure that column values are available, you can use the `FETCHCOLS` or `FETCHCOLSEXCEPT` option of the `TABLE` parameter to fetch the values from the database if they are not present in the log. As an alternative to fetching columns, you can enable supplemental logging for those columns.

Additional SQLEXEC Guidelines

Observe the following `SQLEXEC` guidelines:

- Up to 20 stored procedures or queries can be executed per `TABLE` or `MAP` entry. They execute in the order listed in the parameter statement.
- A database login by the Oracle GoldenGate user must precede the `SQLEXEC` clause. Use the `SOURCEDB` and `USERIDALIAS` parameter in the Extract parameter file or the `TARGETDB` and `USERIDALIAS` parameter in the Replicat parameter file, as needed for the database type and configured authentication method.
- The SQL is executed by the Oracle GoldenGate user. This user must have the privilege to execute stored procedures and call RDBM-supplied procedures.
- Database operations within a stored procedure or query are committed in same context as the original transaction.
- Do not use `SQLEXEC` to update the value of a primary key column. If `SQLEXEC` is used to update the value of a key column, then the Replicat process will not be able to perform a subsequent update or delete operation, because the original key value will be unavailable. If a key value must be changed, you can map the original key value to another column and then specify that column with the `KEYCOLS` option of the `TABLE` or `MAP` parameter.
- For Db2, Oracle GoldenGate uses the ODBC `SQLExecDirect` function to execute a SQL statement dynamically. This means that the connected database server must be able to prepare the statement dynamically. ODBC prepares the SQL statement every time it is executed (at the requested interval). Typically, this does not present a problem to Oracle GoldenGate users. See the IBM Db2 documentation for more information.
- All object names in a `SQLEXEC` statement must be fully qualified with their two-part or three-part names, as appropriate for the database.
- All objects that are affected by a `SQLEXEC` 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 happen before `SQLEXEC` executes.
- 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 or delete the object before the `SQLEXEC` procedure or query executes on it.

Set up and Use the Master Keys and Encryption Keys

You can set the master keys and encryption keys using the **Key Management** tab in the **Configuration** page of the Administration Server.

Using Master Keys

If you want to encrypt your data, then create a Master Key by clicking the + sign in the Master Key section. The master key is generated automatically.

You can change the status of the key to Available or Unavailable, by clicking the edit icon in the Master Key table. You can also delete the Master Key from the table by clicking the delete icon.

For details on the Master Key concept, see *Encrypting Data with the Master Key and Wallet Method*.

Using the Encryption Keys

To use this method of data encryption, you configure Oracle GoldenGate to generate an encryption key and store the key in a local `ENCKEYS` file. The `ENCKEYS` file must be secured through the normal method of assigning file permissions in the operating system. This procedure generates an AES encryption key and provides instructions for storing it in the `ENCKEYS` file.

To generate the `ENCKEYS` files, click the + sign in the Encryption Keys section. The Encryption Keys is generated.

For details on the Encryption Keys concept, see the *Encrypting the Data with the ENCKEYS Method*.

Access the Parameter Files

The Global parameters, Extract, Replicat parameter files are available in the Parameter Files section of the Administration Server.

You use the Administration Server Configuration page and Parameter Files tab to work with your various parameter files.

You use the different parameter file options:

1. Select the **Configuration** option from the Administration Server left-navigation pane.
2. Select the **Parameter Files** tab.
A list of existing parameter files is displayed along with the GLOBALS parameter file.
3. If you select any of the parameter files, you are presented with the option to edit or delete the selected file. If you want to change the GLOBALS parameter file, you need to stop and restart all of the services.
4. Click + add parameter files.
5. Enter the file name and the required parameters. Make sure to enter the file name with the `.prm` extension.
6. Click **Submit**. The new parameter file is displayed in the list of parameter files.

The actual location of the parameter files on the disk can be determined using the following step:

1. Identify the GoldenGate Deployment ETC Home:
 - a. Go to Service Manager Overview page.
 - b. Click the deployment from the Deployments section for which you need to find the parameters file.
 - c. Under the Deployment Detail window, navigate to the Oracle GoldenGate deployment /etc home directory.
 - d. Go into the /config/ogg directory where the parameter file is located.

The following example shows how to navigate to your parameter file location:

```
[oracle ~]$ cd /opt/app/oracle/gg_deployments/Atlanta/etc
[oracle etc]$ cd conf/ogg[oracle ogg]$ lsEXT_DEMO.prm GLOBALS REP_DEMO.prm
```

Configure an Encryption Profile

Oracle GoldenGate Administration Server provides options to set up profiles for managed Extract and Replicat (ER) processes. These processes are assigned auto-start and auto-restart properties to control their life cycles.

You can create profiles for managed processes using the Administration Server or the Admin Client. To create a profile in the Administration Server, perform the following tasks:

1. Click Profile from the Administration Server navigation pane.
2. In the Managed Process Settings tab, you can click + sign to start creating a profile. There's also a default profile preset on this page.
3. Enter the details for the profile options including the Profile Name, Description, Auto Start and Auto Restart options. See the following table for Auto Start and Auto Restart options

Option	Description	Extract Type
Intent	What you want the Extract to be used for, such as High Availability or the Unidirectional default.	Classic, Integrated, and Initial Load
Begin	How you want the Extract to start. At a custom time that you select, a database CSN, or the Now default.	Classic and Integrated
Trail Name	A two character trail name.	Classic and Integrated
Trail Subdirectory, Size, Sequence, and Offset	You can further configure the trail details.	Classic and Integrated
Remote	Set if the trail is not on the same server.	Classic and Integrated
Thread Number	Set to a specific redo log number. The default is 1.	Classic
Encryption Profile	Provide the name of the encryption profile for the Extract. If no encryption profile is created, then the default encryption profile is selected, by default	Classic, Integrated, and Initial Load.

Option	Description	Extract Type
Encryption Profile Type	Provide the type of Key Management Service being used. Oracle Key Vault is selected by default.	Classic, Integrated, and Initial Load.
Managed Options	X	X
Profile Name	Provides the name of the autostart and autorestart profile. You can select the default or custom options.	Classic, Integrated, and Initial Load.
Critical to deployment health	Enable this option if the profile is critical for the deployment health.	Classic, Integrated, and Initial Load.
Auto Start	Enables autostart for the process.	Enables autostart for the process.
Max Retries	Specify the maximum number of retries to try to start the process	Classic, Integrated, and Initial Load.
Retry Delay	Delay time in trying to start the process	Classic, Integrated, and Initial Load.
Retries Window	The duration interval to try to start the process	Classic, Integrated, and Initial Load.
Restart on Failure only	If true the task is only restarted if it fails	Classic, Integrated, and Initial Load.
Disable Task After Retries Exhausted	If true then the task is disabled after exhausting all attempts to restart the process.	Classic, Integrated, and Initial Load.

Access Extract and Replicat Log Information

The diagnosis of Extract and Replicat transactions provides information about the severity of a transaction along with the timestamp. This information is helpful in case you need to determine if and when a particular issue occurred including the cause of the issue.

The Extract and Replicat log information is available on the Diagnosis page of Administration Server. To access the Diagnosis page, click the **left navigation page** of the Administration Server and select **Diagnosis**.

Using the Table

An updated log of Extract and Replicat server messages is displayed. You can sort the list by date or severity by clicking on the adjacent arrow. Also, you can refresh this log and choose how many pages you want to view.

To search, you select Date, Severity, or Message, and then select the appropriate options to construct your search.

Notice the **Notifications** tab at the bottom of the page. It displays server messages, which are not updated in the log due to transaction errors. For example, failure to log in to the database using the database credentials.

Mapping and Manipulating Data

Learn about tasks, functions, commands, and processes used for integrating data between source and target tables.

Guidelines for Using Self-describing Trails

Self-describing trail files are the default trail file format. Oracle recommends that you use self-describing trail files. You should only use `SOURCEDEFS OVERRIDE` and `TARGETDEFS OVERRIDE` for backward compatibility with trail file formats *lower* than 12.2.

If using the self-describing trails, then the column names on the source are mapped to the column names in the target table. Order of columns doesn't matter and if column names are different, then they need to be explicitly mapped using `COLMAP`.

Parameters that Control Mapping and Data Integration

All data selection, mapping, and manipulation that Oracle GoldenGate performs is accomplished by using one or more options of the `TABLE` and `MAP` parameters.

- Use `TABLE` in the Extract parameter file.
- Use `MAP` in the Replicat parameter file.

`TABLE` and `MAP` specify the database objects that are affected by the other parameters in the parameter file. See [Specifying Object Names in Oracle GoldenGate Input](#) for instructions for specifying object names in these parameters.

Mapping between Dissimilar Databases

Mapping and conversion between tables that have different data structures requires either a source-definitions file, a target-definitions file, or in some cases both. Mapping between dissimilar databases is controlled by the self-describing trails, and mapping is done by column name, regardless of the data type for the source or target column.

If you don't want automatic mapping based on the self-describing trails or want backward compatibility then you can use `SOURCEDEFS` or `TARGETDEFS`.

Mapping and Conversion on NonStop Systems

If you are mapping or converting data from a Windows or UNIX system to a NonStop Enscribe target, the mapping or conversion must be performed on the Windows or UNIX source system. Replicat for NonStop cannot convert three-part or two-part SQL table names and data types to the three-part file names that are used for the Enscribe platform. Extract can format the trail data with Enscribe names and target data types.

Mapping and Conversion on Windows and UNIX Systems

When Oracle GoldenGate is operating only on Windows-based and UNIX-based systems, column mapping and conversion can be performed in the Extract process, or in the Replicat process. To prevent the added overhead of this processing on the Extract process, you can configure the mapping and conversion to be performed on the Replicat process or on an intermediary system.

In the case where there are multiple sources and one target, it might be more efficient to perform the mapping and conversion on the source.

Globalization Considerations when Mapping Data

When planning to map and convert data between databases and platforms, take into consideration what is supported or not supported by Oracle GoldenGate in terms of globalization.

Conversion between Character Sets

Oracle GoldenGate converts between source and target character sets if they are different, so that object names and column data are compared, mapped, and manipulated properly from one database to another. See [Supported Character Sets](#), for a list of supported character sets.

To ensure accurate character representation from one database to another, the following must be true:

- The character set of the target database must be a superset or equivalent of the character set of the source database. *Equivalent* means not equal, but having the same set of characters. For example, Shift-JIS and EUC-JP technically are not completely equal, but have the same characters in most cases.
- If your client applications use different character sets, the database character set must also be a superset or equivalent of the character sets of the client applications.
- In many databases, including Oracle, it is possible to force a character into a database that is not part of the Character Set. Oracle GoldenGate considers this as an invalid value, and may not map this character correctly when replicating data. For these types of situations you can use the `REPLACEBADCHAR` parameter as described in the *Parameters and Functions Reference for Oracle GoldenGate*.

In this configuration, every character is represented when converting from a client or source character set to the local database character set.

A Replicat process can support conversion from one source character set to one target character set.

Database Object Names

Oracle GoldenGate processes catalog, schema, table and column names in their native language as determined by the character set encoding of the source and target databases. This support preserves single-byte and multibyte names, symbols, accent characters, and case-sensitivity with locale taken into account where available, at all levels of the database hierarchy.

Column Data

Oracle GoldenGate supports the conversion of column data between character sets when the data is contained in the following column types:

- **Character-type columns:** `CHAR/VARCHAR/CLOB` to `CHAR/VARCHAR/CLOB` of another character set; and `CHAR/VARCHAR/CLOB` to and from `NCHAR/NVARCHAR/NCLOB`.
- Columns that contain string-based numbers and date-time data. Conversions of these columns is performed between z/OS EBCDIC and non-z/OS ASCII data. Conversion is not performed between ASCII and ASCII versions of this data, nor between EBCDIC and EBCDIC versions, because the data are compatible in these cases.

 **Note:**

Oracle GoldenGate supports timestamp data from 0001-01-03 00:00:00 to 9999-12-31 23:59:59. If a timestamp is converted from GMT to local time, these limits also apply to the resulting timestamp. A value of zero month, zero day field, or an all zero date value isn't supported. For example, values such as 0000-00-00 00:00:00, or any date value that includes a zero month or zero day field isn't supported.

Character-set conversion for column data is limited to a direct mapping of a source column and a target column in the `COLMAP` or `USEDEFAULTS` clauses of the Replicat `MAP` parameter. A direct mapping is a name-to-name mapping without the use of a stored procedure or column-conversion function. Replicat performs the character-set conversion. No conversion is performed by Extract.

Preservation of Locale

Oracle GoldenGate takes the locale of the database into account when comparing case-insensitive object names. See [Supported Locales](#) for a list of supported locales.

Support for Escape Sequences

Oracle GoldenGate supports the use of an escape sequence to represent a string column, literal text, or object name in the parameter file. You can use an escape sequence if the operating system does not support the required character, such as a control character, or for any other purpose that requires a character that cannot be used in a parameter file.

An escape sequence can be used anywhere in the parameter file, but is particularly useful in the following elements within a `TABLE` or `MAP` statement:

- An object name
- `WHERE` clause
- `COLMAP` clause to assign a Unicode character to a Unicode column, or to assign a native-encoded character to a column.
- Oracle GoldenGate column conversion functions within a `COLMAP` clause.

Oracle GoldenGate supports the following types of escape sequence:

- `\uFFFF` Unicode escape sequence. Any `UNICODE` code point can be used except surrogate pairs.
- `\377` Octal escape sequence
- `\xFF` Hexadecimal escape sequence

The following rules apply:

- If used for mapping of an object name in `TABLE` or `MAP`, no restriction apply. For example, the following `TABLE` specification is valid:

```
TABLE schema."\u3000ABC";
```

- If used with a column-mapping function, any code point can be used, but only for an `NCHAR`/`NVARCHAR` column. For an `CHAR`/`VARCHAR` column, the code point is limited to the equivalent of 7-bit ASCII.
- The source and target data types must be identical (for example, `NCHAR` to `NCHAR`).

- Begin each escape sequence with a reverse solidus (code point U+005C), followed by the character code point. (A solidus is more commonly known as the backslash symbol.) Use the escape sequence, instead of the actual character, within your input string in the parameter statement or column-conversion function.

 **Note:**

To specify an actual backslash in the parameter file, specify a double backslash. For example, the following finds a backslash in COL1: @STRFIND (COL1, '\\\').

To Use the \uFFFF Unicode Escape Sequence

- The \uFFFF Unicode escape sequence must begin with a lowercase u, followed by exactly four hexadecimal digits.
- Supported ranges are as follows:
 - 0 to 9 (U+0030 to U+0039)
 - A to F (U+0041 to U+0046)
 - a to f (U+0061 to U+0066)

\u20ac is the Unicode escape sequence for the Euro currency sign.

 **Note:**

For reliable cross-platform support, use the Unicode escape sequence. Octal and hexadecimal escape sequences are not standardized on different operating systems.

To Use the \377 Octal Escape Sequence

- Must contain exactly three octal digits.
- Supported ranges:
 - Range for first digit is 0 to 3 (U+0030 to U+0033)
 - Range for second and third digits is 0 to 7 (U+0030 to U+0037)

\200 is the octal escape sequence for the Euro currency sign on Microsoft Windows

To Use the \xFF Hexadecimal Escape Sequence

- Must begin with a lowercase x followed by exactly two hexadecimal digits.
- Supported ranges:
 - 0 to 9 (U+0030 to U+0039)
 - A to F (U+0041 to U+0046)
 - a to f (U+0061 to U+0066)

\x80 is the hexadecimal escape sequence for the Euro currency sign on Microsoft Windows 1252 Latin1 code page.

Mapping Columns Using TABLE and MAP

Oracle GoldenGate provides for column mapping at the table level and at the global level. Default column mapping is also provided in the absence of explicit column mapping rules.

This section contains the following guidelines for mapping columns:

Supporting Case and Special Characters in Column Names

By default, Oracle GoldenGate follows SQL-92 rules for specifying column names and literals. In Oracle GoldenGate parameter files, conversion functions, user exits, and commands, case-sensitive column names must be enclosed within double quotes if double quotes are required by the database to enforce case-sensitivity. For other case-sensitive databases that do not require quotes, case-sensitive column names must be specified as they are stored in the database. Literals must be enclosed within single quotes. See [Differentiating Case-Sensitive Column Names from Literals](#) for more information.

Configuring Table-level Column Mapping with COLMAP

If you are using self-describing trails then any column on the source object is mapped to the same column name on the target object. You only need to manage column names that are different between source and target or if you need to transform a column.

However, if not using self-describing trails then the default mapping is done by column order and not the column name. So column 1 on the source will be mapped to column 1 on the target, column 2 to column 2 and so on.

Use the `COLMAP` option of the `MAP` and `TABLE` parameters to:

- map individual source columns to target columns that have different names.
- specify default column mapping when an explicit column mapping is not needed.
- Provide instructions for selecting, mapping, translating, and moving data from a source column into a target column.

Using USEDEFAULTS to Enable Default Column Mapping

You can use the `USEDEFAULTS` option of `COLMAP` to specify automatic default column mapping for any corresponding source and target columns that have identical names. `USEDEFAULTS` can save you time by eliminating the need to map every target column explicitly.

Default mapping causes Oracle GoldenGate to map those columns and, if required, translate the data types based on the data-definitions file. Do not specify default mapping for columns that are mapped already with an explicit mapping statement.

The following example of a column mapping illustrates the use of both default and explicit column mapping for a source table `ACCTBL` and a target table `ACCTTAB`. Most columns are the same in both tables, except for the following differences:

- The source table has a `CUST_NAME` column, whereas the target table has a `NAME` column.
- A ten-digit `PHONE_NO` column in the source table corresponds to separate `AREA_CODE`, `PHONE_PREFIX`, and `PHONE_NUMBER` columns in the target table.
- Separate `YY`, `MM`, and `DD` columns in the source table correspond to a single `TRANSACTION_DATE` column in the target table.

To address those differences, `USEDEFAULTS` is used to map the similar columns automatically, while explicit mapping and conversion functions are used for dissimilar columns.

The following sample shows the column mapping using the `COLMAP` option of the `MAP` and `TABLE` parameters. It describes the mapping of the source table `ACCTBL` to the target table `ACCTTAB`.

```
MAP SALES.ACCTBL, TARGET SALES.ACCTTAB,
      COLMAP (  USEDEFAULTS,
                NAME = CUST_NAME,
                TRANSACTION_DATE = @DATE ('YYYY-MM-DD', 'YY', YEAR,
'MM', MONTH, 'DD', DAY),
                AREA_CODE = @STREXT (PHONE_NO, 1, 3),
                PHONE_PREFIX = @STREXT (PHONE_NO, 4, 6),
                PHONE_NUMBER = @STREXT (PHONE_NO, 7, 10)
      )
;
```

Table 9-2 Sample Column Mapping

Parameter statement	Description
<code>COLMAP</code>	Begins the <code>COLMAP</code> statement.
<code>USEDEFAULTS,</code>	Maps source columns as-is when the target column names are identical.
<code>NAME = CUST_NAME,</code>	Maps the source column <code>CUST_NAME</code> to the target column <code>NAME</code> .
<code>TRANSACTION_DATE = @DATE ('YYYY-MM-DD', 'YY', YEAR, 'MM', MONTH, 'DD', DAY),</code>	Converts the transaction date from the source date columns to the target column <code>TRANSACTION_DATE</code> by using the <code>@DATE</code> column conversion function.
<code>AREA_CODE = @STREXT (PHONE_NO, 1, 3), PHONE_PREFIX = @STREXT (PHONE_NO, 4, 6), PHONE_NUMBER = @STREXT (PHONE_NO, 7, 10)) ;</code>	Converts the source column <code>PHONE_NO</code> into the separate target columns of <code>AREA_CODE</code> , <code>PHONE_PREFIX</code> , and <code>PHONE_NUMBER</code> by using the <code>@STREXT</code> column conversion function.

See [Understanding Default Column Mapping](#) for more information about the rules followed by Oracle GoldenGate for default column mapping.

Specifying the Columns to be Mapped in the COLMAP Clause

The COLMAP syntax is the following:

```
COLMAP ([USEDEFAULTS, ] target_column = source_expression)
```

In this syntax, *target_column* is the name of the target column and *source_expression*. Some examples of *source_expressions* are:

- The name of a source column, such as `ORD_DATE`.
- Numeric constant, such as `123`.
- String constant enclosed within single quotes, such as `'ABCD'`.
- An expression using an Oracle GoldenGate column-conversion function. Within a COLMAP statement, you can use any of the Oracle GoldenGate column-conversion functions to transform data for the mapped columns, for example:

```
@STREXT (COL1, 1, 3)
```

- Here's an example of using `BEFORE column_name:BEFORE ORD_DATE`
- Here's an example of using `AFTER column_name :AFTER ORD_DATE`. This is the default option if a column name is listed.

If the column mapping involves case-sensitive columns from different database types, specify each column as it is stored in the database.

- If the database requires double quotes to enforce case-sensitivity, specify the case-sensitive column name within double quotes.
- If the database is case-sensitive without requiring double quotes, specify the column name as it is stored in the database.

The following shows a mapping between a target column in an Oracle database and a source column in a case-sensitive SQL Server database.

```
COLMAP ("ColA" = ColA)
```

See [Specifying Object Names in Oracle GoldenGate Input](#) for more information about specifying names to Oracle GoldenGate.

See [Globalization Considerations when Mapping Data](#) for globalization considerations when mapping source and target columns in databases that have different character sets and locales.

Avoid using COLMAP to map a value to a key column (which causes the operation to become a primary key update). The WHERE clause that Oracle GoldenGate uses to locate the target row will not use the correct before image of the key column. Instead, it will use the after image. This will cause errors if you are using any functions based on that key column, such as a `SQLEXEC` statement.

Column Mapping Limitations

Here are the column mapping limitations:

- LOB columns cannot be used in `FILTER`, `WHERE` clauses, or as a `source_expression` in a `COLMAP` statement. LOB columns are BLOB, CLOB, NCLOB, XMLType, User-Defined Data Types, Nested Tables, VARRAYs and other special data types.
- If the source column contains more than 4000 bytes, it cannot be used in transformation routines, as the value is stored in the trail as an LOB record. For example a `VARCHAR2(4000 CHAR)` in Oracle and the Japanese character set is stored as 3 bytes for each character. This implies that the column could be 12000 bytes long and Oracle GoldenGate would store this value as an LOB field.
- The full SQL statement that Oracle GoldenGate would execute would exceed 4MB in size. For example, if you have a table with thousands of `VARCHAR2(4000)` columns and you want to put 4000 bytes in each one, this could cause the total SQL statement that Oracle GoldenGate is going to execute to exceed the maximum size of 4MB.

Configuring Global Column Mapping with COLMATCH

Use the `COLMATCH` parameter to create global rules for column mapping. With `COLMATCH`, you can map between similarly structured tables that 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 table-level mapping with a `COLMAP` clause in individual `TABLE` or `MAP` statements.

Case-sensitivity is supported as follows:

- For MySQL, SQL Server if the database is case-sensitive, `COLMATCH` looks for an exact case and name match regardless of whether or not a name is specified in quotes.
- For Oracle Database and Db2 databases, where names can be either case-sensitive or case-insensitive in the same database and double quotes are required to show case-sensitivity, `COLMATCH` requires an exact case and name match when a name is in quotes in the database.

Syntax

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

Argument	Description
<code>NAMES target_column = source_column</code>	<p>Maps based on column names.</p> <p>Put double quotes around the column name if it is case-sensitive and the database requires quotes to enforce case-sensitivity. For these database types, an unquoted column name is treated as case-insensitive by Oracle GoldenGate.</p> <p>For databases that support case-sensitivity without requiring quotes, specify the column name as it is stored in the database.</p> <p>If the <code>COLMATCH</code> is between columns in different database types, make certain the names reflect the appropriate case representation for each one. For example, the following specifies a case-sensitive target column name "aBc" in an Oracle Database and a case-sensitive source column name aBc in a case-sensitive SQL Server database.</p> <pre>COLMATCH NAMES "aBc" = aBc</pre>
<code>PREFIX prefix SUFFIX suffix</code>	<p>Ignores the specified name prefix or suffix.</p> <p>Put double quotes around the prefix or suffix if the database requires quotes to enforce case-sensitivity, for example "P_". For those database types, an unquoted prefix or suffix is treated as case-insensitive.</p> <p>For databases that support case-sensitivity without requiring quotes, specify the prefix or suffix as it is stored in the database. For example, P_ specifies a capital P prefix.</p> <p>The following example specifies a case-insensitive prefix to ignore. The target column name P_ABC is mapped to source column name ABC, and target column name P_abc is mapped to source column name abc.</p> <pre>COLMATCH PREFIX p_</pre> <p>The following example specifies a case-sensitive suffix to ignore. The target column name ABC_k is mapped to the source column name ABC, and the target column name "abc_k" is mapped to the source column name "abc".</p> <pre>SUFFIX "_k"</pre>
<code>RESET</code>	<p>Turns off previously defined <code>COLMATCH</code> rules for subsequent <code>TABLE</code> or <code>MAP</code> statements.</p>

The following example illustrates when to use `COLMATCH`. The source and target tables are identical except for slightly different table and column names. The database is case-insensitive.

ACCT Table	ORD Table
CUST_CODE	CUST_CODE
CUST_NAME	CUST_NAME
CUST_ADDR	ORDER_ID
PHONE	ORDER_AMT
S_REP	S_REP
S_REPCODE	S_REPCODE

ACCOUNT Table	ORDER Table
CUSTOMER_CODE	CUSTOMER_CODE
CUSTOMER_NAME	CUSTOMER_NAME
CUSTOMER_ADDRESS	ORDER_ID
PHONE	ORDER_AMT
REP	REP
REPCODE	REPCODE

To map the source columns to the target columns in this example, as well as to handle subsequent maps for other tables, the syntax is:

```
COLMATCH NAMES CUSTOMER_CODE = CUST_CODE
COLMATCH NAMES CUSTOMER_NAME = CUST_NAME
COLMATCH NAMES CUSTOMER_ADDRESS = CUST_ADDR
COLMATCH PREFIX S_
MAP SALES.ACCT, TARGET SALES.ACCOUNT, COLMAP (USEDEFAULTS);
MAP SALE.ORD, TARGET SALES.ORDER, COLMAP (USEDEFAULTS);
COLMATCH RESET
MAP SALES.REG, TARGET SALE.REG;
MAP SALES.PRICE, TARGET SALES.PRICE;
```

Based on the rules in the example, the following occurs:

- Data is mapped from the `CUST_CODE` columns in the source `ACCT` and `ORD` tables to the `CUSTOMER_CODE` columns in the target `ACCOUNT` and `ORDER` tables.
- The `s_` prefix will be ignored.
- Columns with the same names, such as the `PHONE` and `ORDER_AMT` columns, are automatically mapped by means of `USEDEFAULTS` without requiring explicit rules. See [Understanding Default Column Mapping](#) for more information.
- The previous global column mapping is turned off for the tables `REG` and `PRICE`. Source and target columns in those tables are automatically mapped because all of the names are identical.

Understanding Default Column Mapping

For self-describing trails, if an explicit column mapping does not exist, either by using `COLMATCH` or `COLMAP`, Oracle GoldenGate maps source and target columns by default according to the following rules.

This doesn't apply if you are using `SOURCEDEFS` or `TARGETDEFS`.

- If a source column is found whose name and case exactly match those of the target column, the two are mapped.
- If no case match is found, fallback name mapping is used. Fallback mapping performs a case-insensitive target table mapping to find a name match. Inexact column name matching is applied using upper cased names. This behavior is controlled by the `GLOBALS` parameter `NAMEMATCHIGNORECASE`. You can disable fallback name matching with the `NAMEMATCHEXACT` parameter, or you can keep it enabled but with a warning message by using the `NAMEMATCHNOWARNING` parameter.
- Target columns that do not correspond to any source column take default values determined by the database.

If the default mapping cannot be performed, the target column defaults to one of the values shown in the following table.

Column Type	Value
Numeric	Zero (0)
Character or <code>VARCHAR</code>	Spaces
Date or Datetime	Current date and time
Columns that can take a <code>NULL</code> value	Null

Data Type Conversions

Learn about how Oracle GoldenGate maps data types.

Numeric Columns

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

You can specify a substitution value for invalid numeric data encountered when mapping number columns by using the `REPLACEBADNUM` parameter for more information.

Character-type Columns

Character-type columns can accept character-based data types such as `VARCHAR`, numeric in string form, date and time in string form, and string literals. If the scale of the target column is smaller than that of the source, the column is truncated on the right. If the scale of the target column is larger than that of the source, the column is padded with spaces on the right.

Literals must be enclosed within single quotes.

You can control the response of the Oracle GoldenGate process when a valid code point does not exist for either the source or target character set when mapping character columns by using the `REPLACEBADCHAR` parameter for more information.

Datetime Columns

Datetime (`DATE`, `TIME`, and `TIMESTAMP`) columns can accept datetime and character columns, as well as string literals. Literals must be enclosed within single quotes. To map a character

column to a datetime column, make certain it conforms to the Oracle GoldenGate external SQL format of `YYYY-MM-DD HH:MI:SS.FFFFFFFF`.

Oracle GoldenGate supports timestamp data from `0001-01-03 00:00:00` to `9999-12-31 23:59:59`. If a timestamp is converted from GMT to local time, these limits also apply to the resulting timestamp. Depending on the timezone, conversion may add or subtract hours, which can cause the timestamp to exceed the lower or upper supported limit.

Required precision varies according to the data type and target platform. If the scale of the target column is smaller than that of the source, data is truncated on the right. If the scale of the target column is larger than that of the source, the column is extended on the right with the values for the current date and time.

Selecting and Filtering Rows

Filtering can only be performed on columns that are available to Oracle GoldenGate. In the `TRANLOG Extract` Oracle GoldenGate has access to all columns that are present in the redo logs and in the database. If the columns are not in the redo logs, they must be explicitly fetched (using `FETCHCOLS`) to be able to filter them. In the `Extract` pump and in the `Replicat`, the columns must be available in the trail file. Because of this, any column that you want to use in a `FILTER` or `WHERE` clause must be explicitly logged using `ADD TRANDATA COLS`, and you have to retain the default of `LOGALLSUPCOLS`.

To filter out or select rows for extraction or replication, use the `FILTER` and `WHERE` clauses of the `TABLE` and `MAP` parameters.

The `FILTER` clause offers you more functionality than the `WHERE` clause because you can employ any of the Oracle GoldenGate column conversion functions, whereas the `WHERE` clause accepts basic `WHERE` operators.

Selecting Rows with a FILTER Clause

Use a `FILTER` clause to select rows based on a numeric value by using basic operators or one or more Oracle GoldenGate column-conversion functions.



Note:

To filter a column based on a string, use one of the Oracle GoldenGate string functions or use a `WHERE` clause.

The syntax for `FILTER` in a `TABLE` statement is as follows:

```
TABLE source_table,
, FILTER (
[, ON INSERT | ON UPDATE | ON DELETE]
[, IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE]
, filter_clause);
```

The syntax for `FILTER` in a `MAP` statement is as follows and includes an error-handling option.

```
MAP source_table, TARGET target_table,
, FILTER (
[, ON INSERT | ON UPDATE | ON DELETE]
```

```
[, IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE]
[, RAISEERROR error_number]
, filter_clause);
```

Valid `FILTER` clause elements are the following:

- An Oracle GoldenGate column-conversion function. These functions are built into Oracle GoldenGate so that you can perform tests, manipulate data, retrieve values, and so forth. See [Testing and Transforming Data](#) for more information about Oracle GoldenGate conversion functions.
- 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`

Use the following `FILTER` options to specify which SQL operations a filter clause affects. Any of these options can be combined.

```
ON INSERT | ON UPDATE | ON DELETE IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE
```

Use the `RAISEERROR` option of `FILTER` in the `MAP` parameter to generate a user-defined error when the filter fails. This option is useful when you need to trigger an event in response to the failure.

Use the `@RANGE` function within a `FILTER` clause to distribute the processing workload among multiple `MAP` or `TABLE` statements.

Here's a sample:

```
REPERROR (9999, EXCEPTION)
MAP OWNER.SRCTAB, TARGET OWNER.TARGETAB,
```

```

        SQLEXEC (ID CHECK, ON UPDATE, QUERY ' SELECT COUNT FROM
TARGTAB WHERE PKCOL = :P1 ', PARAMS (P1 = PKCOL)),
        FILTER (BALANCE > 15000),
        FILTER (ON UPDATE, @BEFORE (COUNT) = CHECK.COUNT)
;
MAP OWNER.SRCTAB, TARGET OWNER.TARGEXC,
EXCEPTIONSONLY,
COLMAP ( USEDEFAULTS,
ERRTYPE = 'UPDATE FILTER FAILED'
)
;

```

Table 9-3 Using Multiple FILTER Statements

Parameter file	Description
REPERROR (9999, EXCEPTION)	Raises an exception for the specified error.
MAP OWNER.SRCTAB, TARGET OWNER.TARGTAB,	Starts the MAP statement.
SQLEXEC (ID CHECK, ON UPDATE, QUERY ' SELECT COUNT FROM TARGTAB ' 'WHERE PKCOL = :P1 ', PARAMS (P1 = PKCOL)),	Performs a query to retrieve the present value of the COUNT column whenever an update is encountered. There is a BEFOREFILTER option also that allows the query or stored procedure to be executed prior to processing the FILTER clause. This allows values from the SQLEXEC portion to be used inside the FILTER at runtime.
FILTER (BALANCE > 15000),	Uses a FILTER clause to select rows where the balance is greater than 15000.
FILTER (ON UPDATE, @BEFORE (COUNT) = CHECK.COUNT)	Uses another FILTER clause to ensure that the value of the source COUNT column before an update matches the value in the target column before applying the target update.
;	The semicolon concludes the MAP statement.
MAP OWNER.SRCTAB, TARGET OWNER.TARGEXC, EXCEPTIONSONLY, COLMAP (USEDEFAULTS, ERRTYPE = 'UPDATE FILTER FAILED');	Designates an exceptions MAP statement. The REPERROR clause for error 9999 ensures that the exceptions map to TARGEXC will be executed.

Example 9-5 Calling the @COMPUTE Function

The following example calls the @COMPUTE function to extract records in which the price multiplied by the amount exceeds 10,000.

```
MAP SALES.TCUSTORD, TARGET SALES.TORD,  
FILTER (@COMPUTE (PRODUCT_PRICE * PRODUCT_AMOUNT) > 10000);
```

Example 9-6 Calling the @STREQ Function

The following uses the @STREQ function to extract records where the value of a character column is 'JOE'.

```
TABLE ACCT.TCUSTORD, FILTER (@STREQ ("Name", 'joe') > 0);
```

Example 9-7 Selecting Records

The following selects records in which the AMOUNT column is greater than 50 and executes the filter on UPDATE and DELETE operations.

```
TABLE ACT.TCUSTORD, FILTER (ON UPDATE, ON DELETE, AMOUNT > 50);
```

Example 9-8 Using the @RANGE Function

(Replicat group 1 parameter file)

```
MAP sales.acct, TARGET sales.acct, FILTER (@RANGE (1, 2, ID));
```

(Replicat group 2 parameter file)

```
MAP sales.acct, TARGET sales.acct, FILTER (@RANGE (2, 2, ID));
```

You can combine several FILTER clauses in one MAP or TABLE statement, as shown in [Table 9-3](#), which shows part of a Replicat parameter file. Oracle GoldenGate executes the filters in the order listed, until one fails or until all are passed. If one filter fails, they all fail.

Selecting Rows with a WHERE Clause

Use any of the elements in [Table 9-4](#) in a WHERE clause to select or exclude rows (or both) based on a conditional statement. Each WHERE clause must be enclosed within parentheses. Literals must be enclosed within single quotes.

Table 9-4 Permissible WHERE Operators

Element	Examples
Column names	PRODUCT_AMT
Numeric values	-123, 5500.123

Table 9-4 (Cont.) Permissible WHERE Operators

Element	Examples
Literal strings	'AUTO', 'Ca'
Built-in column tests	@NULL, @PRESENT, @ABSENT (column is null, present or absent in the row). These tests are built into Oracle GoldenGate. See Considerations for Selecting Rows with FILTER and WHERE .
Comparison operators	=, <>, >, <, >=, <=
Conjunctive operators	AND, OR
Grouping parentheses	Use open and close parentheses () for logical grouping of multiple elements.

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.

Arithmetic operators and floating-point data types are not supported by `WHERE`. To use more complex selection conditions, use a `FILTER` clause or a user exit routine.

The syntax for `WHERE` is identical in the `TABLE` and `MAP` statements:

```
TABLE table, WHERE (clause);
```

```
MAP source_table, TARGET target_table, WHERE (clause);
```

Considerations for Selecting Rows with FILTER and WHERE

The following suggestions can help you create a successful selection clause.



Note:

The examples in this section assume a case-insensitive database.

Ensuring Data Availability for Filters

If the database only logs values for *changed* columns to the transaction log, there can be errors if any of the unchanged columns are referenced by selection criteria. Oracle GoldenGate ignores such row operations, outputs them to the discard file, and issues a warning.

To avoid missing-column errors, create your selection conditions as follows:

- Use only primary-key columns as selection criteria, if possible.
- Make required column values available by enabling supplemental logging for those columns. Alternatively, you can use the `FETCHCOLS` or `FETCHCOLSEXCEPT` option of the `TABLE` parameter. These options are valid for all supported databases. They query the database

to fetch the values if they are not present in the log. To retrieve the values before the `FILTER` or `WHERE` clause is executed, include the `FETCHBEFOREFILTER` option in the `TABLE` statement before the `FILTER` or `WHERE` clause. For example:

```
TABLE DEMO.PEOPLE, FETCHBEFOREFILTER, FETCHCOLS (age), FILTER (age > 50);
```

- Test for a column's presence first, then for the column's value. To test for a column's presence, use the following syntax.

```
column_name {= | <>} {@PRESENT | @ABSENT}
```

The following 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)
```

Comparing Column Values

To ensure that elements used in a comparison match, compare appropriate column types:

- Character columns to literal strings.
- Numeric columns to numeric values, which can include a sign and decimal point.
- Date and time columns to literal strings, using the format in which the column is retrieved by the application.

Testing for NULL Values

To evaluate columns for `NULL` values, use the following syntax.

```
column {= | <>} @NULL
```

The following returns `TRUE` if the column value is `NULL`, and thereby replicates the row. It returns `FALSE` for all other cases (including a column missing from the record).

```
WHERE (amount = @NULL)
```

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

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



Note:

If a value in the trail contains more than 4000 bytes then the `@NULL` function will return `TRUE`.

Retrieving Before and After Values

For update and delete operations, it can be useful to retrieve the `BEFORE` values of the source columns (the values before the update occurred). For inserts, all column values are considered `AFTER` images.

These values are stored in the trail and can be used in filters and column mappings. For example, you can:

- Retrieve the before image of a row as part of a column-mapping specification in an exceptions `MAP` statement, and map those values to an exceptions table for use in testing or troubleshooting conflict resolution routines.

- Perform delta calculations. For example, if a table has a `Balance` column, you can calculate the net result of a particular transaction by subtracting the original balance from the new balance, as in the following example:

```
MAP "owner"."src", TARGET "owner"."targ",
COLMAP (PK1 = PK1, delta = balance - @BEFORE (balance));
```

Note:

The previous example indicates a case-sensitive database such as Oracle. The table names are in quote marks to reflect case-sensitivity.

To Reference the Before Value

1. Use the `@BEFORE` column conversion function with the name of the column for which you want a before value, as follows:

```
@BEFORE (column_name)
```

2. Use the `GETUPDATEBEFORES` parameter in the Extract parameter file to capture before images from the transaction record, or use it in the Replicat parameter file to use the before image in a column mapping or filter. If 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 the values of columns that changed, using the `@BEFORE` function may result in a "column missing" condition and the column map is executed as if the column were not in the record. See [Ensuring Data Availability for Filters](#) to ensure that column values are available.

Oracle GoldenGate also provides the `@AFTER` function to retrieve after values when needed for filtering, for use in conversion functions, or other purposes. See `@BEFORE` and `@AFTER` in the *Parameters and Functions Reference for Oracle GoldenGate*.

Selecting Columns

To control which columns of a source table are extracted by Oracle GoldenGate, use the `COLS` and `COLSEXCEPT` options of the `TABLE` parameter. Use `COLS` to select columns for extraction, and use `COLSEXCEPT` to select all columns except those designated by `COLSEXCEPT`.

Restricting the columns that are extracted can be useful when a target table does not contain the same columns as the source table, or when the columns contain sensitive information, such as a personal identification number or other proprietary business information.

Selecting and Converting SQL Operations

By default, Oracle GoldenGate captures and applies `INSERT`, `UPDATE`, and `DELETE` operations. You can use the following parameters in the Extract or Replicat parameter file to control which kind of operations are processed, such as only inserts or only inserts and updates.

```
GETINSERTS | IGNOREINSERTS
```

```
GETUPDATES | IGNOREUPDATES
```

```
GETDELETES | IGNOREDELETES
```

You can convert one type of SQL operation to another by using the following parameters in the Replicat parameter file:

- Use `INSERTUPDATES` to convert source update operations to inserts into the target table. This is useful for maintaining a transaction history on that table. The transaction log record must contain all of the column values of the table, not just changed values. Some databases do not log full row values to their transaction log, but only values that changed.
- Use `INSERTDELETES` to convert all source delete operations to inserts into the target table. This is useful for retaining a history of all records that were ever in the source database.
- Use `UPDATEDELETES` to convert source deletes to updates on the target.

Using Transaction History

Oracle GoldenGate enables you to retain a history of changes made to a target record and to map information about the operation that caused each change. This history can be useful for creating a transaction-based reporting system that contains a separate record for every operation performed on a table, as opposed to containing only the most recent version of each record.

For example, the following series of operations made to a target table named `CUSTOMER` would leave no trace of the ID of `Dave`. The last operation deletes the record, so there is no way to find out Dave's account history or his ending balance.

Table 9-5 Operation History for Table `CUSTOMER`

Sequence	Operation	ID	BALANCE
1	Insert	Dave	1000
2	Update	Dave	900
3	Update	Dave	1250
4	Delete	Dave	1250

Retaining this history as a series of records can be useful in many ways. For example, you can generate the net effect of transactions.

To Implement Transaction Reporting

1. To prepare Extract to capture before values, use the `GETUPDATEBEFORES` parameter in the Extract parameter file. A before value (or before image) is the existing value of a column before an update is performed. Before images enable Oracle GoldenGate to create the transaction record.
2. To prepare Replicat to post all operations as inserts, use the `INSERTALLRECORDS` parameter in the Replicat parameter file. Each operation on a table becomes a new record in that table.
3. To map the transaction history, use the return values of the `GGHEADER` option of the `@GETENV` column conversion function. Include the conversion function as the source expression in a `COLMAP` statement in the `TABLE` or `MAP` parameter.

Using the sample series of transactions shown in [Table 9-5](#) the following parameter configurations can be created to generate a more transaction-oriented view of customers, rather than the latest state of the database.

Process	Parameter statements
Extract	<pre>GETUPDATEBEFORES TABLE ACCOUNT.CUSTOMER;</pre>
Replicat	<pre>INSERTALLRECORDS MAP SALES.CUSTOMER, TARGET SALES.CUSTHIST, COLMAP (TS = @GETENV ('GGHEADER', 'COMMITTIMESTAMP'), BEFORE_AFTER = @GETENV ('GGHEADER', 'BEFOREAFTERINDICATOR'), OP_TYPE = @GETENV ('GGHEADER', 'OPTYPE'), ID = ID, BALANCE = BALANCE);</pre>

**Note:**

This is not representative of a complete parameter file for an Oracle GoldenGate process. Also note that these examples represent a case-insensitive database.

This configuration makes possible queries such as the following, which returns the net sum of each transaction along with the time of the transaction and the customer ID.

```
SELECT AFTER.ID, AFTER.TS, AFTER.BALANCE - BEFORE.BALANCE
FROM CUSTHIST AFTER, CUSTHIST BEFORE
WHERE AFTER.ID = BEFORE.ID AND AFTER.TS = BEFORE.TS AND
AFTER.BEFORE_AFTER = 'A' AND BEFORE.BEFORE_AFTER = 'B';
```

Testing and Transforming Data

Data testing and transformation can be performed by either Extract or Replicat and is implemented by using the Oracle GoldenGate built-in column-conversion functions within a `COLMAP` clause of a `TABLE` or `MAP` statement. With these conversion functions, you can:

- Transform dates.
- Test for the presence of column values.
- Perform arithmetic operations.
- Manipulate numbers and character strings.
- Handle null, invalid, and missing data.
- Perform tests.

If you need to use logic beyond that which is supplied by the Oracle GoldenGate functions, you can call your own functions by implementing Oracle GoldenGate user exits.

Oracle GoldenGate conversion functions take the following general syntax:

Syntax

@function (argument)

Table 9-6 Conversion Function Syntax

Syntax element	Description
<i>@function</i>	The Oracle GoldenGate function name. Function names have the prefix @, as in @COMPUTE or @DATE. A space between the function name and the open-parenthesis before the input argument is optional.
<i>argument</i>	A function argument.

Table 9-7 Function Arguments

Argument element	Example
A numeric constant	123
A string literal enclosed within single quote marks	'ABCD'
The name of a source column	PHONE_NO or phone_no, or "Phone_No" or Phone_no Depends on whether the database is case-insensitive, is case-sensitive and requires quote marks to enforce the case, or is case-sensitive and does not require quotes.
An arithmetic expression	COL2 * 100
A comparison expression	((COL3 > 100) AND (COL4 > 0))
Other Oracle GoldenGate functions	AMOUNT = @IF (@COLTEST (AMT, MISSING, INVALID), 0, AMT)

Handling Column Names and Literals in Functions

By default, literal strings must be enclosed in single quotes in a column-conversion function. Case-sensitive column names must be enclosed within double quotes if required by the database, or otherwise entered in the case in which they are stored in the database.

Using the Appropriate Function

Use the appropriate function for the type of column that is being manipulated or evaluated. For example, numeric functions can be used only to compare numeric values. To compare character values, use one of the Oracle GoldenGate character-comparison functions. LOB columns cannot be used in conversion functions.

This statement would fail because it uses @IF, which is a numerical function, to compare string values.

```
@IF (SR_AREA = 'Help Desk', 'TRUE', 'FALSE')
```

The following statement would succeed because it compares a numeric value.

```
@IF (SR_AREA = 20, 'TRUE', 'FALSE')
```

See [Manipulating Numbers and Character Strings](#) for more information.



Note:

Errors in argument parsing sometimes are not detected until records are processed. Verify syntax before starting processes.

Transforming Dates

Use the @DATE, @DATEDIF, and @DATENOW functions to retrieve dates and times, perform computations on them, and convert them.

This example computes the time that an order is filled

Example 9-9 Computing Time

```
ORDER_FILLED = @DATE (
  'YYYY-MM-DD HH:MI:SS',
  'JTS',
  @DATE ('JTS',
  'YYMMDDHHMISS',
  ORDER_TAKEN_TIME) +
  ORDER_MINUTES * 60 * 1000000)
```

Performing Arithmetic Operations

To return the result of an arithmetic expression, use the @COMPUTE function. The value returned from the function is in the form of a string. 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.
```

Omitting @COMPUTE

The `@COMPUTE` keyword is not required when an expression is passed as a function argument.

```
@STRNUM ((AMOUNT1 + AMOUNT2), LEFT)
```

The following expression returns the same result as the previous one:

```
@STRNUM (@COMPUTE (AMOUNT1 + AMOUNT2), LEFT)
```

Manipulating Numbers and Character Strings

To convert numbers and character strings, Oracle GoldenGate supplies the following functions:

Table 9-8 Conversion Functions for Numbers and Characters

Purpose	Conversion Function
Convert a binary or character string to a number.	@NUMBIN @NUMSTR
Convert a number to a string.	@STRNUM
Compare strings.	@STRCMP @STRNCMP

Table 9-8 (Cont.) Conversion Functions for Numbers and Characters

Purpose	Conversion Function
Concatenate strings.	@STRCAT
	@STRNCAT
Extract from a string.	@STREXT
	@STRFIND
Return the length of a string.	@STRLEN
Substitute one string for another.	@STRSUB
Convert a string to upper case.	@STRUP
Trim leading or trailing spaces, or both.	@STRLTRIM
	@STRRTRIM
	@STRTRIM

Handling Null, Invalid, and Missing Data

When column data is missing, invalid, or null, an Oracle GoldenGate conversion function returns a corresponding value.

If **BALANCE** is 1000, but **AMOUNT** is NULL, the following expression returns NULL:

```
NEW_BALANCE = @COMPUTE (BALANCE + AMOUNT)
```

These exception conditions render the entire calculation invalid. To ensure a successful conversion, use the @COLSTAT, @COLTEST and @IF functions to test for, and override, the exception condition.

Using @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.

The following example returns a NULL into target column **ITEM**.

```
ITEM = @COLSTAT (NULL)
```

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))
```

Using @COLTEST

Use the @COLTEST function to check for the following conditions:

- **PRESENT** tests whether a column is present and not null.
- **NULL** tests whether a column is present and null.
- **MISSING** tests whether a column is not present.
- **INVALID** tests whether a column is present but contains invalid data.

The following example checks whether the `AMOUNT` column is present and `NULL` and whether it is present but invalid.

```
@COLTEST (AMOUNT, NULL, INVALID)
```

Using @IF

Use the `@IF` function to return one of two values based on a condition. Use it with the `@COLSTAT` and `@COLTEST` functions to begin a conditional argument that tests for one or more exception conditions and then directs processing based on the results of the test.

```
NEW_BALANCE = @IF (@COLTEST (BALANCE, NULL, INVALID) OR  
@COLTEST (AMOUNT, NULL, INVALID), @COLSTAT (NULL), BALANCE + AMOUNT)
```

This conversion returns one of the following:

- `NULL` when `BALANCE` or `AMOUNT` is `NULL` or `INVALID`
- `MISSING` when either column is missing
- The sum of the columns.

Performing Tests

The `@CASE`, `@VALONEOF`, and `@EVAL` functions provide additional methods for performing tests on data before manipulating or mapping it.

Using @CASE

Use `@CASE` to select a value depending on a series of value tests.

```
@CASE (PRODUCT_CODE, 'CAR', 'A car', 'TRUCK', 'A truck')
```

This example returns the following:

- `A car` if `PRODUCT_CODE` is `CAR`
- `A truck` if `PRODUCT_CODE` is `TRUCK`
- `A FIELD_MISSING` indication if `PRODUCT_CODE` fits neither of the other conditions

Using @VALONEOF

Use `@VALONEOF` to compare a column or string to a list of values.

```
@IF (@VALONEOF (STATE, 'CA', 'NY'), 'COAST', 'MIDDLE')
```

In this example, if `STATE` is `CA` or `NY`, the expression returns `COAST`, which is the response returned by `@IF` when the value is non-zero (meaning `TRUE`).

Using @EVAL

Use `@EVAL` to select a value based on a series of independent conditional tests.

```
@EVAL (AMOUNT > 10000, 'high amount', AMOUNT > 5000, 'somewhat high')
```

This example returns the following:

- `high amount` if `AMOUNT` is greater than 10000
- `somewhat high` if `AMOUNT` is greater than 5000, and less than or equal to 10000, (unless the prior condition was satisfied)

- A `FIELD_MISSING` indication if neither condition is satisfied.

Using Tokens

You can capture 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 information.

For example, you can use token data in:

- Column maps
- Stored procedures called by a `SQLEXEC` statement
- User exits
- Macros

Defining Tokens

To use tokens, you define the token name and associate it with data. The data can be any valid character data or values retrieved from Oracle GoldenGate column-conversion functions.

The token area in the record header permits up to 16,000 bytes of data. Token names, the length of the data, and the data itself must fit into that space.

To define a token, use the `TOKENS` option of the `TABLE` parameter in the Extract parameter file.

Syntax

```
TABLE table_spec, TOKENS (token_name = token_data [, ...]);
```

Where:

- *table_spec* is the name of the source table. A container or catalog name, if applicable, and an owner name must precede the table name.
- *token_name* is a name of your choice for the token. It can be any number of alphanumeric characters and is not case-sensitive.
- *token_data* is a character string of up to 2000 bytes. The data can be either a string that is enclosed within single quotes or the result of an Oracle GoldenGate column-conversion function. 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. In the trail file, user tokens are stored in UTF-8.

```
TABLE ora.oratest, TOKENS (  
TK-OSUSER = @GETENV ('GGENVIRONMENT' , 'OSUSERNAME'),  
TK-GROUP = @GETENV ('GGENVIRONMENT' , 'GROUPNAME')  
TK-HOST = @GETENV ('GGENVIRONMENT' , 'HOSTNAME'));
```

As shown in this example, the Oracle GoldenGate `@GETENV` function is an effective way to populate token data. This function provides several options for capturing environment information that can be mapped to tokens and then used on the target system for column mapping.

Using Token Data in Target Tables

To map token data to a target table, use the `@TOKEN` column-conversion function in the source expression of a `COLMAP` clause in a Replicat `MAP` statement. The `@TOKEN` function provides the name of the token to map. The `COLMAP` syntax with `@TOKEN` is:

Syntax

```
COLMAP (target_column = @TOKEN ('token_name'))
```

The following MAP statement maps target columns `host`, `gg_group`, and so forth to tokens `tk-host`, `tk-group`, and so forth. Note that the arguments must be enclosed within single quotes.

User tokens	Values
tk-host	:sysA
tk-group	:extora
tk-osuser	:jad
tk-domain	:admin
tk-ba_ind	:B
tk-commit_ts	:2011-01-24 17:08:59.000000
tk-pos	:3604496
tk-rba	:4058
tk-table	:oratest
tk-optype	:insert

Example 9-10 MAP Statement

```
MAP ora.oratest, TARGET ora.rpt,  
COLMAP (USEDEFAULTS,  
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'));
```

The tokens in this example will look similar to the following within the record header in the trail:

Bi-Directional Replication

In a bi-directional configuration, there are Extract and Replicat processes on both the source and target systems to support the replication of transactional changes on each system to the other system. To support this configuration, each Extract must be able to filter the transactions applied by the local Replicat, so that they are not recaptured and sent back to their source in a

continuous loop. Additionally, `AUTO_INCREMENT` columns must be set so that there is no conflict between the values on each system.

Prerequisites for Bidirectional Replication

Learn about the requirements that you must fulfill before you configure a bidirectional replication.

Enable Bi-Directional Loop Detection

Loop detection is a requirement for bi-directional implementations of Oracle GoldenGate, so that an Extract for one source database does not recapture transactions sent by a Replicat from another source database.

With the CDC Extract capture method, by default, any transaction committed by a Replicat into a database where an Extract is configured, will recapture that transaction from the Replicat as long as supplemental logging is enabled for those tables that the Replicat is delivering to.

In order to ignore recapturing transactions that are applied by a Replicat, you must use the `TRANLOGOPTIONS EXCLUDEFILERTABLE` parameter for the CDC Extract. The table used as the filtering table will be the Oracle GoldenGate checkpoint table that you must create for the Replicat.



Note:

Only Classic and Coordinated Replicats support bi-directional and multi-directional replication, parallel Replicat does not support this.

To create a filter table and enable supplemental logging:

1. On each source database, ensure that a checkpoint table for use by Replicats has been created. For example:

```
ADD CHECKPOINTTABLE ggadmin.oggcheck
```

2. Enable supplemental logging for the the checkpoint table. For example:

```
ADD TRANDATA ggadmin.ggcheckpoint ALLCOLS
```

3. Ensure that the Replicat is created with the checkpoint table information.

```
ADD REPLICAT reptgt1, EXTTRAIL /north/e2, CHECKPOINTTABLE ggadmin.ggcheckpoint
```

4. Configure each Extract with the `EXCLUDEFILERTABLE` parameter, using the Replicat's checkpoint table for the filtering table.

```
TRANLOGOPTIONS EXCLUDEFILERTABLE ggadmin.ggcheckpoint
```



Note:

Oracle GoldenGate for PostgreSQL supports only one `EXCLUDEFILERTABLE` statement per Extract, so for multi-directional implementations, ensure each Replicat uses the same checkpoint table in the database that they deliver to.

Considerations for an Active-Active Configuration

The following considerations apply in an active-active configuration. In addition, review the Oracle GoldenGate installation and configuration document for your type of database to see if there are any other limitations or requirements to support a bi-directional configuration.

Application Design

When using Active-Active replication, the time zones must be the same on both systems so that timestamp-based conflict resolution and detection can operate.

Active-active replication is not recommended for use with commercially available packaged business applications, unless the application is designed to support it. Among the obstacles that these applications present are:

- Packaged applications might contain objects and data types that are not supported by Oracle GoldenGate.
- They might perform automatic DML operations that you cannot control, but which will be replicated by Oracle GoldenGate and cause conflicts when applied by Replicat.
- You probably cannot control the data structures to make modifications that are required for active-active replication.

Keys

For accurate detection of conflicts, all records must have a unique, not-null identifier. If possible, create a primary key. If that is not possible, use a unique key or create a substitute key with a `KEYCOLS` option of the `MAP` and `TABLE` parameters. In the absence of a unique identifier, Oracle GoldenGate uses all of the columns that are valid in a `WHERE` clause, but this will degrade performance if the table contains numerous columns.

To maintain data integrity and prevent errors, the following must be true of the key that you use for any given table:

- contain the same columns in all of the databases where that table resides.
- contain the same values in each set of corresponding rows across the databases.

Database-Generated Values

Do not replicate database-generated sequential values, such as Oracle sequences, in a bi-directional configuration. The range of values must be different on each system, with no chance of overlap. For example, in a two-database environment, you can have one server generate even values, and the other odd. For an n -server environment, start each key at a different value and increment the values by the number of servers in the environment. This method may not be available to all types of applications or databases. If the application permits, you can add a location identifier to the value to enforce uniqueness.

Database Configuration

One of the databases must be designated as the *trusted source*. This is the primary database and its host system from which the other database is derived in the initial synchronization phase and in any subsequent resynchronizations that become necessary. Maintain frequent backups of the trusted source data.

Preventing Data Looping

In a bidirectional configuration, SQL changes that are replicated from one system to another must be prevented from being replicated back to the first system. Otherwise, it moves back and forth in an endless loop, as in this example:

1. A user application updates a row on system A.
2. Extract extracts the row on system A and sends it to system B.
3. Replicat updates the row on system B.
4. Extract extracts the row on system B and sends it back to system A.
5. The row is applied on system A (for the second time).
6. This loop continues endlessly.

To prevent data loopback, you may need to provide instructions that:

- prevent the capture of SQL operations that are generated by Replicat, but enable the capture of SQL operations that are generated by business applications if they contain objects that are specified in the Extract parameter file.
- identify local Replicat transactions, in order for the Extract process to ignore them.

Identifying Replicat Transactions

To configure Extract to identify Replicat transactions, follow the instructions for the database from which Extract will capture data.

DB2 z/OS

Identify the Replicat user name by using the following parameter statement in the Extract parameter file.

```
TRANLOGOPTIONS EXCLUDEUSER user
```

This parameter statement marks all DDL and DML transactions that are generated by this user as Replicat transactions. The user name is included in the transaction record that is read by Extract.

MySQL

Identify the name of the Replicat checkpoint table by using the following parameter statement in the Extract parameter file.

```
TRANLOGOPTIONS EXCLUDEFILTERTABLE table_name
```

Replicat writes a checkpoint to the checkpoint table at the end of each of its transactions as part of its checkpoint procedure. (This is the table that is created with the `ADD CHECKPOINTTABLE` command.) Because every Replicat transaction includes a write to this table, it can be used to identify Replicat transactions in a bidirectional configuration. `EXCLUDEFILTERTABLE` identifies the name of the checkpoint table, so that Extract ignores transactions that contain any operations on it.

PostgreSQL and SQL Server

Identify the name of the Replicat checkpoint table by using the following parameter statement in the Extract parameter file and ensure that the Replicat checkpoint table has been enabled for supplemental logging with the `ADD TRANDATA` command.

```
TRANLOGOPTIONS EXCLUDEFILTERTABLE table_name
```

Replicat writes a checkpoint to the checkpoint table at the end of each of its transactions as part of its checkpoint procedure. (This is the table that is created with the `ADD CHECKPOINTTABLE` command). Because every Replicat transaction includes a write to this table, it can be used to identify Replicat transactions in a bi-directional configuration. `EXCLUDEFILTERTABLE` identifies the name of the checkpoint table, so that Extract ignores transactions that contain any operations on it.

Oracle

There are multiple ways to identify Replicat transaction in an Oracle environment. When Replicat is in classic or integrated mode, you use the following parameters:

- Replicats set a tag of 00 by default. Use `DBOPTIONS` with the `SETTAG` option in the Replicat parameter file to change the tag that Replicat sets. Replicat tags the transactions being applied with the specified value, which identifies those transactions in the redo stream. Valid values are a single TAG value consisting of hexadecimal digits.
- Use the `TRANLOGOPTIONS` parameter with the `EXCLUDETAG` option in the Extract parameter file. The logmining server associated with that Extract excludes redo that is tagged with the `SETTAG` value.

The following shows how `SETTAG` can be set in the Replicat parameter file:

```
DBOPTIONS SETTAG 0935
```

The following shows how `EXCLUDETAG` can be set in the Extract parameter file:

```
TRANLOGOPTIONS EXCLUDETAG 0935
```

If you are excluding multiple tags, each must have a separate `TRANLOGOPTIONS EXCLUDETAG` statement specified.

You can also use the transaction name or USERID of the Replicat user to identify Replicat transactions. You can choose which of these to ignore when you configure Extract.

Preventing the Capture of Replicat Operations

Depending on which database you are using, you may or may not need to provide explicit instructions to prevent the capture of Replicat operations.

Oracle: Preventing the Capture of Replicat Transactions

To prevent the capture of SQL that is applied by Replicat to an Oracle database, use the `TRANLOGOPTIONS` parameter with the `EXCLUDETAG tag` option. This parameter directs the Extract process to ignore transactions that are tagged with the specified redo tag.

See [Identifying Replicat Transactions](#) to set the tag value. This is the recommended approach for Oracle.

Non-Oracle Database: Preventing Capture of Replicat Transactions

To prevent the capture of SQL that is applied by Replicat to other database types, use the following parameters:

- `GETAPPLPLOS | IGNOREAPPLPLOS`: Controls whether or not data operations (DML) produced by business applications *except Replicat* are included in the content that Extract writes to a specific trail or file.
- `GETREPLICATES | IGNOREREPLICATES`: Controls whether or not DML operations produced by *Replicat* are included in the content that Extract writes to a specific trail or file.

Manage Conflicts

Uniform conflict-resolution procedures must be in place on all systems in an active-active configuration. Conflicts should be identified immediately and handled with as much automation as possible; however, different business applications will present their own unique set of requirements in this area.

Because Oracle GoldenGate is an asynchronous solution, conflicts can occur when modifications are made to identical sets of data on separate systems at (or almost at) the same time. Conflicts occur when the timing of simultaneous changes results in one of these out-of-sync conditions:

- A **uniqueness conflict** occurs when Replicat applies an insert or update operation that violates a uniqueness integrity constraint, such as a `PRIMARY KEY` or `UNIQUE` constraint. An example of this conflict type is when two transactions originate from two different databases, and each one inserts a row into a table with the same primary key value.
- An **update conflict** occurs when Replicat applies an update that conflicts with another update to the same row. Update conflicts happen when two transactions that originate from different databases update the same row at nearly the same time. Replicat detects an update conflict when there is a difference between the old values (the before values) that are stored in the trail record and the current values of the same row in the target database.
- A **delete conflict** occurs when two transactions originate at different databases, and one deletes a row while the other updates or deletes the same row. In this case, the row does not exist to be either updated or deleted. Replicat cannot find the row because the primary key does not exist.

For example, UserA on DatabaseA updates a row, and UserB on DatabaseB updates the same row. If UserB's transaction occurs before UserA's transaction is synchronized to DatabaseB, there will be a conflict on the replicated transaction.

A more complicated example involves three databases and illustrates a more complex ordering conflict. Assume three databases A, B, and C. Suppose a user inserts a row at database A, which is then replicated to database B. Another user then modifies the row at database B, and the row modification is replicated to database C. If the row modification from B arrives at database C before the row insert from database A, C will detect a conflict.

Where possible, try to minimize or eliminate any chance of conflict. Some ways to do so are:

- Configure the applications to restrict which columns can be modified in each database. For example, you could limit access based on geographical area, such as by allowing different sales regions to modify only the records of their own customers. As another example, you could allow a customer service application on one database to modify only the `NAME` and `ADDRESS` columns of a customer table, while allowing a financial application on another database to modify only the `BALANCE` column. In each of those cases, there cannot be a conflict caused by concurrent updates to the same record.
- Keep synchronization latency low. If UserA on DatabaseA and UserB on DatabaseB both update the same rows at about the same time, and UserA's transaction gets replicated to the target row before UserB's transaction is completed, conflict is avoided. See [Managing Conflicts](#) for suggestions on improving the performance of the Oracle GoldenGate processes.

To avoid conflicts, replication latency must be kept as low as possible. When conflicts are unavoidable, they must be identified immediately and resolved with as much automation as possible, either through the Oracle GoldenGate Conflict Detection and Resolution (CDR) feature, or through methods developed on your own. Custom methods can be integrated into Oracle GoldenGate processing through the `SQLEXEC` and user exit functionality. See [Manual](#)

[Conflict Detection and Resolution](#) for more information about using Oracle GoldenGate to handle conflicts.

For Oracle database, the automatic Conflict Detection Resolution (CDR) feature exists. To know more, see [Automatic Conflict Detection and Resolution](#).

MySQL: Bi-Directional Replication

In a bidirectional configuration, there are Extract and Replicat processes on both the source and target systems to support the replication of transactional changes on each system to the other system. To support this configuration, each Extract must be able to filter the transactions applied by the local Replicat, so that they are not recaptured and sent back to their source in a continuous loop. Additionally, `AUTO_INCREMENT` columns must be set so that there is no conflict between the values on each system.

1. To filter out Replicat operations in a bi-directional configuration so that the applied operations are not captured and looped back to the source again, take the following steps on each MySQL database:
 - Configure each Replicat process to use a checkpoint table. Replicat writes a checkpoint to this table at the end of each transaction. You can use one global checkpoint table or one per Replicat process. See [Checkpoint Tables Additional Details](#).
 - Specify the name of the checkpoint table with the `EXCLUDEFILTERTABLE` option of the `TRANLOGOPTIONS` parameter in the Extract parameter file. The Extract process will ignore transactions that end with an operation to the specified table, which should only be those of Replicat.

Note:

Although optional for other supported databases as a means of enhancing recovery, the use of a checkpoint table is required for MySQL when using bidirectional replication (and likewise, will enhance recovery).

If using a parallel Replicat in a bidirectional replication, then multiple filter tables are supported using the `TRANLOGOPTIONS EXCLUDEFILTERTABLE` option. Multiple filter tables allow the `TRANLOGOPTIONS EXCLUDEFILTERTABLE` to be specified multiple times with different table names or wildcards.

You can include single or multiple `TRANLOGOPTIONS EXCLUDEFILTERTABLE` entries in the Extract parameter file. In the following example, multiple `TRANLOGOPTIONS EXCLUDEFILTERTABLE` entries are included in the Extract parameter file with explicit object names and wildcards.

```
TRANLOGOPTIONS EXCLUDEFILTERTABLE ggs.chkpt2
TRANLOGOPTIONS EXCLUDEFILTERTABLE ggs.chkpt_RABC_*
```

2. Edit the MySQL server configuration file to set the `auto_increment_increment` and `auto_increment_offset` parameters to avoid discrepancies that could be caused by the bi-directional operations. The following illustrates these parameters, assuming two servers: **ServerA** and **ServerB**.

ServerA:

```
auto-increment-increment = 2
auto-increment-offset = 1
```

ServerB:

```
auto-increment-increment = 2
auto-increment-offset = 2
```

PostgreSQL: Bi-Directional Replication

In a bidirectional configuration, there are Extract and Replicat processes on both the source and target systems to support the replication of transactional changes on each system to the other system. To support this configuration, each Extract must be able to filter the transactions applied by the local Replicat, so that they are not recaptured and sent back to their source in a continuous loop.

1. Configure Oracle GoldenGate for high availability or active-active replication according to the instructions in the [Preparing DBFS for an Active-Active Configuration](#).
2. To filter out Replicat operations in a bi-directional configuration so that the applied operations are not captured and looped back to the source again, take the following steps on each PostgreSQL database:
 - Configure each Replicat process to use a checkpoint table. Replicat writes a checkpoint to this table at the end of each transaction. You can use one global checkpoint table or one per Replicat process.
 - Specify the name of the checkpoint table with the `EXCLUDEFILTERTABLE` option of the `TRANLOGOPTIONS` parameter in the Extract parameter file. The Extract process will ignore transactions that end with an operation to the specified table, which should only be those of Replicat.

If using a parallel Replicat in a bidirectional replication, then multiple filter tables are supported using the `TRANLOGOPTIONS EXCLUDEFILTERTABLE` option. Multiple filter tables allow the `TRANLOGOPTIONS EXCLUDEFILTERTABLE` to be specified multiple times with different table names or wildcards.

You can include single or multiple `TRANLOGOPTIONS EXCLUDEFILTERTABLE` entries in the Extract parameter file. In the following example, multiple `TRANLOGOPTIONS EXCLUDEFILTERTABLE` entries are included in the Extract parameter file with explicit object names and wildcards.

```
TRANLOGOPTIONS EXCLUDEFILTERTABLE ggs.chkpt2
TRANLOGOPTIONS EXCLUDEFILTERTABLE ggs.chkpt_RABC_*
```

Preparing DBFS for an Active-Active Configuration

Learn the steps to configure Oracle GoldenGate to function within an active-active bidirectional or multi-directional environment where Oracle Database File System (DBFS) is in use on both (or all) systems.

Supported Operations and Prerequisites

This topic lists what is supported by Oracle GoldenGate for DBFS.

Oracle GoldenGate for DBFS supports the following:

- Supported DDL (like `TRUNCATE` or `ALTER`) on DBFS objects except for `CREATE` statements on the DBFS objects. `CREATE` on DBFS must be excluded from the configuration, as must any

schemas that will hold the created DBFS objects. The reason to exclude `CREATE` is that the metadata for DBFS must be properly populated in the SYS dictionary tables (which itself is excluded from Oracle GoldenGate capture by default).

- Capture and replication of DML on the tables that underlie the DBFS file system.

Applying the Required Patch

Apply the Oracle DBFS patch for bug-9651229 on both databases.

To determine if the patch is installed, run the following query:

```
connect / as sysdba
select procedure_name
from dba_procedures
where object_name = 'DBMS_DBFS_SFS_ADMIN'
and procedure_name = 'PARTITION_SEQUENCE';
```

The query should return a single row. Anything else indicates that the proper patched version of DBFS is not available on your database.

Examples Used in these Procedures

The following procedures assume two systems and configure the environment so that DBFS users on both systems see the same DBFS files, directories, and contents that are kept in synchronization with Oracle GoldenGate.

It is possible to extend these concepts to support three or more peer systems.

Partitioning the DBFS Sequence Numbers

DBFS uses an internal sequence-number generator to construct unique names and unique IDs.

These steps partition the sequences into distinct ranges to ensure that there are no conflicts across the databases. After this is done, further DBFS operations (both creation of new file systems and subsequent file system operations) can be performed without conflicts of names, primary keys, or IDs during DML propagation.

1. Connect to each database as `sysdba`.

Issue the following query on each database.

```
SELECT LAST_NUMBER
FROM DBA_SEQUENCES
WHERE SEQUENCE_OWNER = 'SYS'
AND SEQUENCE_NAME = 'DBFS_SFS_$FSSEQ'
```

2. From this query, choose the maximum value of `LAST_NUMBER` across both systems, or pick a high value that is significantly larger than the current value of the sequence on either system.
3. Substitute this value ("`maxval`" is used here as a placeholder) in both of the following procedures. These procedures logically index each system as `myid=0` and `myid=1`.

Node1

```
declare begin dbms_dbfs_sfs_admin.partition_sequence(nodes => 2, myid =>
0, newstart => :maxval);
commit; end; /
```

Node 2

```
DECLARE
BEGIN
DBMS_DBFS_SFS_ADMIN.PARTITION_SEQUENCE(NODES => 2, MYID => 0, NEWSTART
=> :MAXVAL);
COMMIT;
END;
/
```

 **Note:**

Notice the difference in the value specified for the `myid` parameter. These are the different index values.

For a multi-way configuration among three or more databases, you could make the following alterations:

- Adjust the maximum value that is set for `maxval` upward appropriately, and use that value on all nodes.
 - Vary the value of `myid` in the procedure from 0 for the first node, 1 for the second node, 2 for the third one, and so on.
4. (Recommended) After (and only after) the DBFS sequence generator is partitioned, create a new DBFS file system on each system, and use only these file systems for DML propagation with Oracle GoldenGate. See [Configuring the DBFS file system](#).

 **Note:**

DBFS file systems that were created before the patch for bug-9651229 was applied or before the DBFS sequence number was adjusted can be configured for propagation, but that requires additional steps not described in this document. If you must retain old file systems, open a service request with Oracle Support.

Configuring the DBFS file system

To replicate DBFS file system operations, use a configuration that is similar to the standard bi-directional configuration for DML.

Some guidelines to follow while configuring Oracle GoldenGate for DBFS are:

- Use matched pairs of identically structured tables.
- Allow each database to have write privileges to opposite tables in a set, and set the other one in the set to read-only. For example:

- Node1 writes to local table `t1` and these changes are replicated to `t1` on Node2.
- Node2 writes to local table `t2` and these changes are replicated to `t2` on Node1.
- On Node1, `t2` is read-only. On Node2, `t1` is read-only.

DBFS file systems make this kind of table pairing simple because:

- The tables that underlie the DBFS file systems have the same structure.
- These tables are modified by simple, conventional DML during higher-level file system operations.
- The DBFS ContentAPI provides a way of unifying the namespace of the individual DBFS stores by means of mount points that can be qualified as read-write or read-only.

The following steps create two DBFS file systems (in this case named `FS1` and `FS2`) and set them to be read-write or read, as appropriate.

1. Run the following procedure to create the two file systems. (Substitute your store names for `FS1` and `FS2`.)
2. Run the following procedure to give each file system the appropriate access rights. (Substitute your store names for `FS1` and `FS2`.)

In this example, note that on Node 1, store `FS1` is read-write and store `FS2` is read-only, while on Node 2 the converse is true: store `FS1` is read-only and store `FS2` is read-write.

Note also that the read-write store is mounted as *local* and the read-only store is mounted as *remote*. This provides users on each system with an identical namespace and identical semantics for read and write operations. Local path names can be modified, but remote path names cannot.

Example 9-11

```
DECLARE
DBMS_DBFS_SFS.CREATEFILE SYSTEM('FS1');
DBMS_DBFS_SFS.CREATEFILE SYSTEM('FS2');

DBMS_DBFS_CONTENT.REGISTERSTORE('FS1',
'POSIX', 'DBMS_DBFS_SFS');
DBMS_DBFS_CONTENT.REGISTERSTORE('FS2',
'POSIX', 'DBMS_DBFS_SFS');
COMMIT;
END;
/
```

Example 9-12 Node 1

```
DECLARE
DBMS_DBFS_CONTENT.MOUNTSTORE('FS1', 'LOCAL');
DBMS_DBFS_CONTENT.MOUNTSTORE('FS2', 'REMOTE',
READ_ONLY => TRUE);
COMMIT;
END;
/
```

Example 9-13 Node 2

```

DECLARE
DBMS_DBFS_CONTENT.MOUNTSTORE('FS1', 'REMOTE',
READ_ONLY => TRUE);
DBMS_DBFS_CONTENT.MOUNTSTORE('FS2', 'LOCAL');
COMMIT;
END;
/

```

Mapping Local and Remote Peers Correctly

The names of the tables that underlie the DBFS file systems are generated internally and dynamically.

Continuing with the preceding example, there are:

- Two nodes (Node 1 and Node 2 in the example).
- Four stores: two on each node (FS1 and FS2 in the example).
- Eight underlying tables: two for each store (a table and a ptable). These tables must be identified, specified in Extract `TABLE` statements, and mapped in Replicat `MAP` statements.

1. To identify the table names that back each file system, issue the following query. (Substitute your store names for FS1 and FS2.)

The output looks like the following examples.

2. Identify the tables that are *locally read-write* to Extract by creating the following `TABLE` statements in the Extract parameter files. (Substitute your pluggable database names, schema names, and table names as applicable.)
3. Link changes on each remote file system to the corresponding local file system by creating the following `MAP` statements in the Replicat parameter files. (Substitute your pluggable database, schema and table names.)

This mapping captures and replicates local read-write *source* tables to remote read-only peer tables:

- file system changes made to FS1 on Node 1 propagate to FS1 on Node 2.
- file system changes made to FS2 on Node 2 propagate to FS2 on Node1.

Changes to the file systems can be made through the DBFS ContentAPI (package `DBMS_DBFS_CONTENT`) of the database or through `dbfs_client` mounts and conventional file systems tools.

All changes are propagated in both directions.

- A user at the virtual root of the DBFS namespace on each system sees identical content.
- For mutable operations, users use the `/local` sub-directory on each system.
- For read operations, users can use either of the `/local` or `/remote` sub-directories, depending on whether they want to see local or remote content.

Example 9-14

```

select fs.store_name, tb.table_name, tb.ptable_name
from table(dbms_dbfs_sfs.listTables) tb,

```

```

table(dbms_dbfs_sfs.listfile systems) fs
where fs.schema_name = tb.schema_name
and fs.table_name = tb.table_name
and fs.store_name in ('FS1', 'FS2')
;

```

Example 9-15 Example output: Node 1 (Your Table Names Will Be Different.)

STORE_NAME	TABLE_NAME	PTABLE_NAME
FS1	SFS\$_FST_100	SFS\$_FSTP_100
FS2	SFS\$_FST_118	SFS\$_FSTP_118

Example 9-16 Example output: Node 2 (Your Table Names Will Be Different.)

STORE_NAME	TABLE_NAME	PTABLE_NAME
FS1	SFS\$_FST_101	SFS\$_FSTP_101
FS2	SFS\$_FST_119	SFS\$_FSTP_119

Example 9-17 Node1

```

TABLE [container.]schema.SFS$_FST_100
TABLE [container.]schema.SFS$_FSTP_100;

```

Example 9-18 Node2

```

TABLE [container.]schema.SFS$_FST_119
TABLE [container.]schema.SFS$_FSTP_119;

```

Example 9-19 Node1

```

MAP [container.]schema.SFS$_FST_119, TARGET [container.]schema.SFS$_FST_118;
MAP [container.]schema.SFS$_FSTP_119, TARGET [container.]schema.SFS$_FSTP_118

```

Example 9-20 Node2

```

MAP [container.]schema.SFS$_FST_100, TARGET [container.]schema.SFS$_FST_101;MAP
[container.]schema.SFS$_FSTP_100, TARGET [container.]schema.SFS$_FSTP_101;

```

Using Edition-Based Redefinition

Oracle GoldenGate supports the use of Edition-based Redefinition (EBR) with Oracle Databases enabling you to upgrade the database component of an application while it is in use, thereby minimizing or eliminating down time.

Editions are non-schema objects that Editioned objects belong to. Editions can be thought of as owning editioned objects or as a namespace. Every database starts with one edition named, `ORA$BASE`; this includes upgraded databases. More than one edition can exist in a database and each can only have one child. For example, if you create three editions in succession, `edition1`, `edition2`, `edition3`, then `edition1` is the parent of `edition2` which is the parent of `edition3`. This is irrespective of the user or database session that creates them or which edition was current when the new one is created. When you create an edition, it inherits all the editioned objects of its parent. To use editions with Oracle GoldenGate, you must create them.

An object is considered editioned if it is an editionable type, it is created with the `EDITIONABLE` attribute, and the schema is enabled for editioning of that object type. When you create, alter, or drop an editioned object, the redo log will contain the name of the edition in which it belongs.

In a container database, editions belong to the container and each container has its own default edition.

The `CREATE` | `DROP EDITION` DDLs are captured for all Extract configurations. They fall into the `OTHER` category and assigned an `OBJTYPE` option value of `EDITION`. The `OBJTYPE` option can be used for filtering, for example:

```
DDL EXCLUDE OBJTYPE EDITION
DDL EXCLUDE OBJTYPE EDITION OPTYPE CREATE
DDL EXCLUDE OBJTYPE EDITION OPTYPE DROP
DDL EXCLUDE OBJTYPE EDITION OPTYPE DROP ALLOWEMPTYOWNER OBJNAME edition_name
```

You must use the following syntax to exclude an edition from Extract or Replicat:

```
EXCLUDE OBJTYPE EDITION, ALLOWEMPTYOWNER OBJNAME edition_name
```

Editions fall into the `OTHER` category so no mapping is performed on the edition name. When applied, the `USE` permission is automatically granted to the Replicat user. Replicat will also perform a `grant use on edition_name with grant option to the original creating user` if that user exists on the target database. Because editions are not mappable operations, they do not have owners so the standard `EXCLUDE` statement does not work.

The DDLs used to create or alter editions do not enable the user for editions, rather they enable the schema for editions. This is an important distinction because it means that the Replicat user does not need to be enabled for editions to apply DDLs to editioned objects. When Replicat applies a `CREATE EDITION` DDL, it grants the original creating user permission to `USE` it, if the original user exists on the target database. For any unreplicated `CREATE EDITION` statements, you must issue a `USE WITH GRANT OPTION` grant to the Replicat user.

Whether or not an editionable objects becomes editioned is controlled by the schema it is applied in. Replicat switches its current session Edition before applying a DDL if the edition name attribute exists in the trail file and it is not empty.

Container database environments are supported for both Extract and Replicat. No additional configuration is necessary. The Replicat user's schema can not be enabled for editions if it is a common user. The Replicat user's schema does not need to be enabled for editions when applying DDLs to editioned objects in other schemas.

Error Management

Learn about configuring the Oracle GoldenGate processes to handle errors.

Oracle GoldenGate reports processing errors in several ways by means of its monitoring and reporting tools.

Also see: [Monitor](#).

Automatic Conflict Detection and Resolution

When Oracle GoldenGate replicates changes between Oracle databases, you can configure and manage Oracle GoldenGate Automatic Conflict Detection and Resolution in the Oracle databases.

**Note:**

The Automatic Conflict Detection and Resolution feature is available from Oracle Database 12c Release 2 (12.2) and later and works with Oracle GoldenGate 12c (12.3.0.1) and later releases. There is a manual conflict detection and resolution feature, which is called Oracle GoldenGate conflict detection and resolution (CDR). Oracle GoldenGate CDR is configured in the Replicat parameter file. To know more about Oracle GoldenGate CDR, see [Manual Conflict Detection and Resolution](#).

About Automatic Conflict Detection and Resolution

When Oracle GoldenGate replicates changes between Oracle databases, you can configure and manage Oracle GoldenGate conflict detection and resolution automatically in these databases.

This feature is intended for use with active-active configurations, where Oracle GoldenGate must maintain data synchronization among multiple databases that contain the same data sets.

**Note:**

Automatic conflict detection and resolution (ACDR) feature that is available only when using Oracle GoldenGate with Oracle Database. For non-Oracle databases, there is a manual conflict detection and resolution (CDR) feature available with Oracle GoldenGate. Oracle GoldenGate CDR is configured in the Replicat parameter file.

Automatic Conflict Detection and Resolution

You can configure automatic conflict detection and resolution in an Oracle GoldenGate configuration that replicates tables between Oracle Databases. To configure conflict detection and resolution for a table, call the `ADD_AUTO_CDR` procedure in the `DBMS_GOLDENGATE_ADM` package.

When Oracle GoldenGate captures changes that originated at an Oracle Database, each change is encapsulated in a row logical change record (LCR). A row LCR is a structured representation of a DML row change. Each row LCR includes the operation type, old column values, and new column values. Multiple row LCRs can be part of a single database transaction.

When more than one replica of a table allows changes to the table, a conflict can occur when a change is made to the same row in two different databases at nearly the same time. Oracle GoldenGate replicates changes using the row LCRs. It detects a conflict by comparing the old values in the row LCR for the initial change from the origin database with the current values of the corresponding table row at the destination database identified by the key columns. If any column value does not match, then there is a conflict.

After a conflict is detected, Oracle GoldenGate can resolve the conflict by overwriting values in the row with some values from the row LCR, ignoring the values in the row LCR, or computing a delta to update the row values.

Automatic conflict detection and resolution does not require application changes for the following reasons:

- Oracle Database automatically creates and maintains invisible timestamp columns.
- Inserts, updates, and deletes use the delete tombstone log table to determine if a row was deleted.
- LOB column conflicts can be detected.
- Oracle Database automatically configures supplemental logging on required columns.

**Note:**

If you use the classic Replicat on tables that have Automatic Change Detection and Resolution enabled, the Extract might abend with the OGG-10461 Failed to retrieve timestamp error. This is because the internal trigger that inserts the records into tombstone tables, only fires on user DMLs. A classic Replicat suppresses all the triggers from firing, which results in missing inserts on tombstone tables.

Requirements for Automatic Conflict Detection and Resolution

Supplemental logging is required to ensure that each row LCR has the information required to detect and resolve a conflict. Supplemental logging places additional information in the redo log for the columns of a table when a DML operation is performed on the table. When you configure a table for Oracle GoldenGate conflict detection and resolution, supplemental logging is configured automatically for all of the columns in the table. The additional information in the redo log is placed in an LCR when a table change is replicated.

Extract must be used for capturing. Integrated Replicat or parallel Replicat in integrated mode must be used on the apply side. `LOGALLSUPCOLS` should remain the default.

There is a hidden field `KEYVER$$` of type timestamp that is optionally added to the `DELETE TOMBSTONE` table. This field is required for `EARLIEST_TIMESTAMP`, `DELETE ALWAYS WINS`, and `SITE PRIORITY` resolution and it also exists in the base table. The existence of the field in the base table needs to be provided in the trail file metadata as a flag or token.

Primary Key updates is also supported in the `DELETE TOMBSTONE` table. An entry is inserted into the `DELETE TOMBSTONE` table for the row of the original key value (before image). The logic in the Extract which matches inserts in the `DELETE TOMBSTONE` table to deletes also needs to be matched to PK updates, or unique key (UK) with at least one non-nullable field, if there is no PK.

Site priority needs support from the Replicat, both the parameters are implemented and the setting is passed to the apply.

Compatibility and Migration

If the base table at the source database does not contain the `KEYVER$$` column, but the target base table has, `DELETE` and Primary Key Updates causes an error at the target database for `EARLIEST_TIMESTAMP`, `DELETE ALWAYS WINS`, and `SITE PRIORITY` resolutions.

When replicating from a base table, which has a `KEYVER$$` to a target table, which does not, the `KEYVER$$` column is ignored.

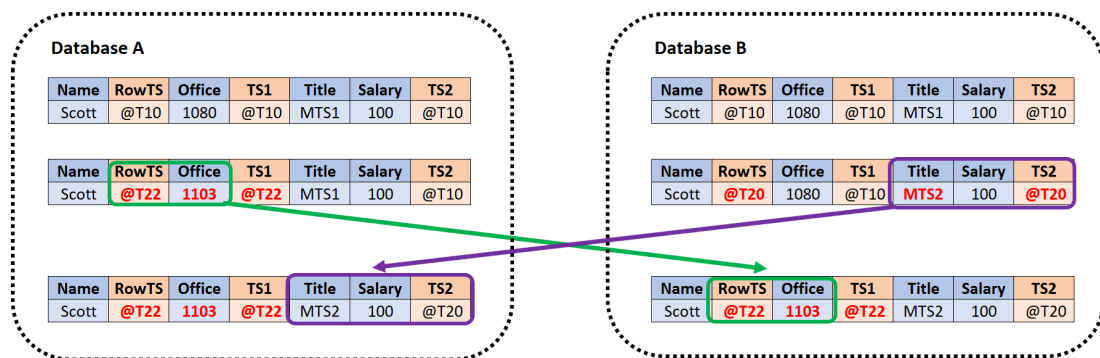
Column Groups

A column group is a logical grouping of one or more columns in a replicated table. When you add a column group, conflict detection and resolution is performed on the columns in the column group separately from the other columns in the table.

When you configure a table for Oracle GoldenGate conflict detection and resolution with the `ADD_AUTO_CDR` procedure, all of the scalar columns in the table are added to a default column group. To define other column groups for the table, run the `ADD_AUTO_CDR_COLUMN_GROUP` procedure. Any columns in the table that are not part of a user-defined column group remain in the default column group for the table.

Column groups enable different databases to update different columns in the same row at nearly the same time without causing a conflict. When column groups are configured for a table, conflicts can be avoided even if different databases update the same row in the table. A conflict is not detected if the updates change the values of columns in different column groups.

Figure 9-1 Column Groups



This example shows a row being replicated at database A and database B. The following two column groups are configured for the replicated table at each database:

- One column group includes the `Office` column. The invisible timestamp column for this column group is `TS1`.
- Another column group includes the `Title` and `Salary` columns. The invisible timestamp column for this column group is `TS2`.

These column groups enable database A and database B to update the same row at nearly the same time without causing a conflict. Specifically, the following changes are made:

- At database A, the value of `Office` was changed from 1080 to 1030.
- At database B, the value of `Title` was changed from `MTS1` to `MTS2`.

Because the `Office` column and the `Title` column are in different column groups, the changes are replicated without a conflict being detected. The result is that values in the row are same at both databases after each change has been replicated.

Piecewise LOB Updates

A set of lob operations composed of `LOB WRITE`, `LOB ERASE`, and `LOB TRIM` is a piecewise LOB update. When a table that contains LOB columns is configured for conflict detection and

resolution, each LOB column is placed in its own column group, and the column group has its own hidden timestamp column. The timestamp column is updated on the first piecewise LOB operation.

For a LOB column, a conflict is detected and resolved in the following ways:

- If the timestamp for the LOB's column group is later than the corresponding LOB column group in the row, then the piecewise LOB update is applied.
- If the timestamp for the LOB's column group is earlier than the corresponding LOB column group in the row, then the LOB in the table row is retained.
- If the row does not exist in the table, then an error is raised.

DELETE TOMBSTONE Table

`DELETE TOMBSTONE` table is a marker for a deleted record to distinguish it from a record, which never existed. A `DELETE TOMBSTONE` table contains at minimum the key columns and operation timestamp. This information is required for delete convergence because some incoming updates and inserts may be delayed from another site and the incoming LCR needs to be filtered against the tombstone operation timestamp to determine whether it should be applied.

Earliest Timestamp Conflict Detection and Resolution

Columns with names of the form `CDRTS$ column group` and `CDRTS$ROW` are used to contain timestamps that reflect modification times for column groups and the row.



Note:

Tables with \$ or \$\$ symbols are internal or hidden tables.

The `DBMS_GOLDENGATE_ADM` includes the following procedures for configuring earliest and latest timestamp resolution:

- `ADD_AUTO_CDR()`
- `ADD_AUTO_CDR_COLUMN_GROUP()`
- `REMOVE_AUTO_CDR()`
- `REMOVE_AUTO_CDR_COLUMN_GROUP()`
- `ALTER_AUTO_CDR()`
- `ALTER_AUTO_CDR_COLUMN_GROUP()`

The field `ADDITIONAL_OPTIONS` in both `ADD_AUTO_CDR()` and `ALTER_AUTO_CDR()` turn on the use of earliest timestamp. Turning on earliest timestamp automatically turn on versioning, which adds a new hidden column `KEYVER$$` (version number) of type timestamp. A new flag value is added to indicate the earliest timestamp usage. This field is also added to the `DELETE TOMBSTONE` table. Delete conflicts are the reason that version number is needed. With an earliest timestamp resolution, delete conflicts, which can be transparent, might not only incorrectly succeed, they might prevent new inserts of the row (new versions). With a version timestamp, the delete can be correctly resolved against a row DML for the same row version.

The original insert of the row receives the current timestamp from its default value. The delete of this row then inserts the version number and the time when this row was inserted, into the

tombstone table when there is a delete. On a new insert, by default, the version number receives the current timestamp again, thereby avoiding a false conflict with the present delete entries in the tombstone table.

Example

Assume that you have a table **tab1** which is globally consistent between databases on site 1 and site 2. The table contains a (primary) key. ACDR is automatically maintaining a key version (**kv**) and timestamp (**ts**) as columns for the base table (hidden) and the tombstone table. For key version **kv** and timestamp **ts**

Database 1: insert tab1 key1 kv1 ts1

Database 2: delete tab1 key1 kv1 ts1

Insertion to DELETE TOMBSTONE table key1 kv1 ts1

Database 1: insert tab1 key1 kv2 ts2

Without using the key version, the insert would be ignored, the delete timestamp is earlier. As the key version is used, you know that **kv2** is not the version of the row that was deleted and the insert succeeds.

Latest Timestamp Conflict Detection and Resolution

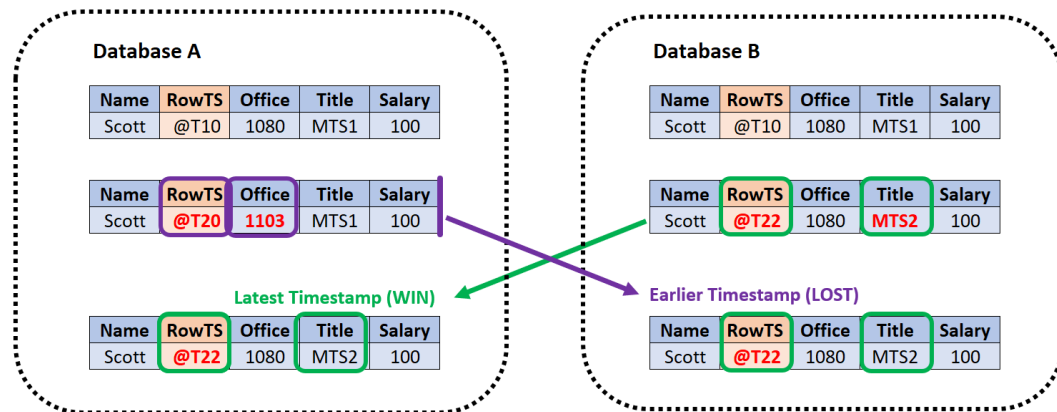
When you run the `ADD_AUTO_CDR` procedure in the `DBMS_GOLDENGATE_ADM` package to configure a table for automatic Oracle GoldenGate conflict detection and resolution, a hidden timestamp column is added to the table. This hidden timestamp column records the time of a row change, and this information is used to detect and resolve conflicts.

When a row LCR is applied, a conflict can occur for an `INSERT`, `UPDATE`, or `DELETE` operation. The following table describes each type of conflict and how it is resolved.

Operation	Conflict Detection	Conflict Resolution
INSERT	A conflict is detected when the table has the same value for a key column as the new value in the row LCR.	<p>If the timestamp of the row LCR is later than the timestamp in the table row, then the values in the row LCR replace the values in the table.</p> <p>If the timestamp of the row LCR is earlier than the timestamp in the table row, then the row LCR is discarded, and the table values are retained.</p>

Operation	Conflict Detection	Conflict Resolution
UPDATE	<p>A conflict is detected in each of the following cases:</p> <ul style="list-style-type: none"> There is a mismatch between the timestamp value in the row LCR and the timestamp value of the corresponding row in the table. There is a mismatch between an old value in a column group in the row LCR does not match the column value in the corresponding table row. A column group is a logical grouping of one or more columns in a replicated table. The table row does not exist. If the row is in the tombstone table, then this is referred to as an update-delete conflict. 	<p>If there is a value mismatch and the timestamp of the row LCR is later than the timestamp in the table row, then the values in the row LCR replace the values in the table.</p> <p>If there is a value mismatch and the timestamp of the row LCR is earlier than the timestamp in the table row, then the row LCR is discarded, and the table values are retained.</p> <p>If the table row does not exist and the timestamp of the row LCR is later than the timestamp in the tombstone table row, then the row LCR is converted from an UPDATE operation to an INSERT operation and inserted into the table.</p> <p>If the table row does not exist and the timestamp of the row LCR is earlier than the timestamp in the tombstone table row, then the row LCR is discarded.</p> <p>If the table row does not exist and there is no corresponding row in the tombstone table, then the row LCR is converted from an UPDATE operation to an INSERT operation and inserted into the table.</p>
DELETE	<p>A conflict is detected in each of the following cases:</p> <ul style="list-style-type: none"> There is a mismatch between the timestamp value in the row LCR and the timestamp value of the corresponding row in the table. The table row does not exist. 	<p>If the timestamp of the row LCR is later than the timestamp in the table, then delete the row from the table.</p> <p>If the timestamp of the row LCR is earlier than the timestamp in the table, then the row LCR is discarded, and the table values are retained.</p> <p>If the delete is successful, then log the row LCR by inserting it into the tombstone table.</p> <p>If the table row does not exist, then log the row LCR by inserting it into the tombstone table.</p>

The following image displays the conflict resolution between database A and database B:



This example shows a row being replicated at database A and database B. The database columns are Name, RowTS, Office, Title, and Salary. The RowTS column is the invisible column in both databases. There is an update in the Office column in database A and at the same time there is a update in the Title column in database B. This causes a conflict and the resolution for this conflict is done applying the latest timestamp method.

- In database A, the value in the Office column gets updated from **1080** to **1103** and the RowTS value changes from **@TS10** to **@TS20**. A arrow indicates that this change is replicated to database B.
- In database B, the value of the Title column changes from **MTS1** to **MTS2** and the RowTS value changes from **@TS10** to **@TS22**.
- To resolve this conflict, the latest timestamp which exists in database B wins. This implies that the changes in database A are not applied. The final values applied to database A and database B are Scott, @TS22, 1080, MTS2, 100.

Delete Always Wins Timestamp CDR

DELETE ALWAYS WINS is enabled through the field `ADDITIONAL_OPTIONS` in both `DBMS_GOLDENGATE_ADM` procedures `ADD_AUTO_CDR()` and `ALTER_AUTO_CDR()`. This is again a delete conflict resolution method, which is not using latest timestamp resolution, therefore, versioning is needed. Turning on `DELETE ALWAYS WINS` automatically turns on versioning, which adds a new hidden column `KEYVER$$` (version number) of type timestamp. A new flag value is also added to `acdrflags_kqldtvc` to indicate `DELETE ALWAYS WINS` usage. This field is also added to the `DELETE TOMBSTONE` table. The same versioning issues exist as the `EARLIEST TIMESTAMP` resolution.

Example:

Key Version `kv` and Timestamp `ts`

Database 1: `insert tab1 key1 kv1 ts1`

Database 2: `delete tab1 key1 kv1 ts1`

Insertion to `DELETE TOMBSTONE` table `key1 kv1 ts1`

Database 1: `insert tab1 key1 kv2 ts2`

Without using the key version, the insert would be ignored, the delete always wins. As the key version is used, you know that `kv2` is not the version of the row that was deleted and the insert succeeds.

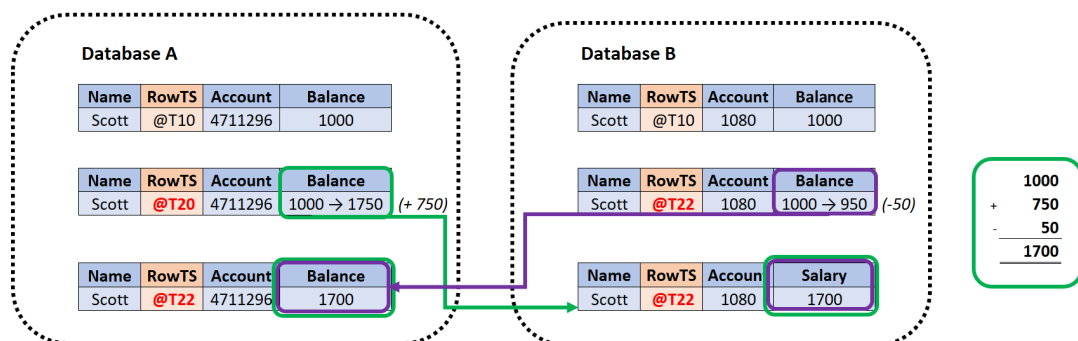
Delta Conflict Resolution

With delta conflict detection, a conflict occurs when a value in the old column list of the row LCR differs from the value for the corresponding row in the table.

To configure delta conflict detection and resolution for a table, run the `ADD_AUTO_CDR_DELTA_RES` procedure in the `DBMS_GOLDENGATE_ADM` package. The delta resolution method does not depend on a timestamp or an extra resolution column. With delta conflict resolution, the conflict is resolved by adding the difference between the new and old values in the row LCR to the value in the table. This resolution method is generally used for financial data such as an account balance. For example, if a bank balance is updated at two sites concurrently, then the converged value accounts for all debits and credits.

The following figure provides an example that illustrates delta conflict detection and resolution.

Figure 9-2 Delta Conflict Detection and Resolution



This example shows a row being replicated at database A and database B. The `Balance` column is designated as the column on which delta conflict resolution is performed, and the `RowTS` column is the invisible timestamp column to track the time of each change to the `Balance` column. A change is made to the `Balance` value in the row in both databases at nearly the same time (`@T20` in database A and `@T22` in database B). These changes result in a conflict, and delta conflict resolution is used to resolve the conflict in the following way:

- At database A, the value of `Balance` was changed from 1000 to 1750. Therefore, the value was increased by 750.
- At database B, the value of `Balance` was changed from 1000 to 950. Therefore, the value was decreased by 50.
- To resolve the conflict at database A, the value of the difference between the new and old values in the row LCR to the value in the table. The difference between the new and old values in the LCR is $(1000 + 750 - 50 = 1700)$. The current value in the table is increased by 700 so that the value after conflict resolution is 1700.
- To resolve the conflict at database B, the value of the difference between the new and old values in the row LCR to the value in the table. The difference between the new and old values in the LCR is 750 $(1000 - 50 + 750) = 1700$. Therefore, the current value in the table (950) is increased by 750 so that the value after conflict resolution is 1700.

After delta conflict resolution, the value of the `Balance` column is the same for the row at database A and database B.

Site Priority CDR



Note:

`SITE PRIORITY` resolution takes precedence over all `COLUMN GROUP` resolution settings.



Note:

If `SITE PRIORITY Replicat` parameter is not placed before applicable map statements in the parameter file, it will not work. This parameter must be placed before the applicable map statements.

Priority resolution is specified in Replicat parameter file between source and target for conflict resolution.

`SITE PRIORITY` is enabled for a database or PDB in the Replicat parameter file with the parameter `ACDR SITE_PRIORITY {source_db_name}{OVERWRITE | IGNORE }`, which is specified to turn on `SITE PRIORITY` resolution for a table.

If the `OVERWRITE` option is specified, then the source table takes priority and conflicts are resolved by `OVERWRITE`. Conversely, if the `IGNORE` option is specified, then the target table takes priority and the source table changes are ignored in a conflict.

`SITE PRIORITY` resolution can be disabled by the field `ADDITIONAL_OPTIONS` in the `ADD_AUTO_CDR()` procedure in `DBMS_GOLDENGATE_ADM` package, and `ALTER_AUTO_CDR()` by setting `IGNORE_SITE_PRIORITY`.

Every Replicat source-target relationship can be set up differently, therefore, convergence is dependent on user setup.

Track Primary Key Updates in Delete Tombstone

Full support of primary key (PK) updates requires handling conflicts on both the rows represented by the before image of the key and the row represented by the after image of the key. A PK update is an autonomous delete and insert, so, the PK update conflicts must be supported as a delete for conflicts with the before image of the key and inserts with the after image of the key (and row).

Supporting the PK update as a delete of the row represented by the before image of the key means that it should insert into the delete tombstone table as a delete. An update internal trigger is added to insert into the tombstone table when the PK is updated (actually the row identifying key, either the PK if it exists or the chosen UK with at least one non-nullable column). As a PK update may lead to two conflicts, up to two resolutions are attempted at the row level, delete of the row with the original PK and the insert of the row with the new PK.

Example: Using latest timestamp resolution

Database 1: Update to `tab1 key1` at `ts1`

Database 2: Update to `tab1 key1` set `key1` to `key2` `ts2`

Database 3: Update to `tab1 key2` `ts3`

In this scenario, it appears that at the row level `tab1` row with `key1` should be deleted and the database 3 update should be the final modification of `tab1` row `key2`. If instead the database 2 is at `ts3` and database 3 is at `ts3`, then the PK update at database 2 would be the final modification of `tab1` row `key2`.

Now, consider a case where the database 1 was at `ts3`, database 2 at `ts2` and database 3 at `ts1`, then the update to `tab1` row `key1` on database 1 should succeed and the PK update from database 2 on `tab1` row `key2` should succeed. At this point, it looks like the complete resolution is that both the delete at the before image and the insert at the after image must be resolved separately. This implies that they are not dependent on each other and a loss for one, is not a loss for both.

Configuring Delta Conflict Detection and Resolution

The `ADD_AUTO_CDR_DELTA_RES` procedure in the `DBMS_GOLDENGATE_ADM` package configures delta conflict detection and resolution.

With delta conflict resolution, you specify one column for which conflicts are detected and resolved. The conflict is detected if the value of the column in the row LCR does not match the corresponding value in the table. The conflict is resolved by adding the difference between the new and old values in the row LCR to the value in the table.

You can configure an Oracle GoldenGate administrator using the `GRANT_ADMIN_PRIVILEGE` procedure in the `DBMS_GOLDENGATE_ADM` package.

1. Connect to the inbound server database as an Oracle GoldenGate administrator.
2. Run the `ADD_AUTO_CDR` procedure and specify the table to configure for latest timestamp conflict detection and resolution.
3. Run the `ADD_AUTO_CDR_DELTA_RES` procedure and specify the column on which delta conflict detection and resolution is performed.
4. Repeat the previous steps in each Oracle Database that replicates the table.

Example 9-21 Configuring Delta Conflict Detection and Resolution for a Table

This example configures delta conflict detection and resolution for the `order_total` column in the `oe.orders` table.

```
BEGIN
  DBMS_GOLDENGATE_ADM.ADD_AUTO_CDR(
    SCHEMA_NAME => 'OE',
    TABLE_NAME => 'ORDERS');
END;
/

BEGIN
  DBMS_GOLDENGATE_ADM.ADD_AUTO_CDR_DELTA_RES(
    SCHEMA_NAME => 'OE',
    TABLE_NAME => 'ORDERS',
    COLUMN_NAME => 'ORDER_TOTAL');
END;
/
```

Configuring Latest Timestamp Conflict Detection and Resolution

The `ADD_AUTO_CDR` procedure in the `DBMS_GOLDENGATE_ADM` package configures latest timestamp conflict detection and resolution. The `ADD_AUTO_CDR_COLUMN_GROUP` procedure adds optional column groups.

For Oracle Database 23ai and higher, additional methods exist to manage and maintain `ACDR` configured tables. You can retain the underlying `AUTO-CDR`-related columns as `UNUSED` columns or drop them immediately after calling the `REMOVE_AUTO_CDR` procedure.

If you apply the `ADD_AUTO_CDR` procedure to a table, then by default, its internal columns are marked as unused if `AUTO_CDR` is removed. After calling `REMOVE_AUTO_CDR`, the unused columns could be manually deleted at a later stage or can be immediately removed using some additional parameters. For details, see [Removing Conflict Detection and Resolution From a Table](#).

To know more, see `ADD_AUTO_CDR` Procedure in the *Oracle Database PL/SQL Packages and Types Reference*

With latest timestamp conflict detection and resolution, a conflict is detected when the timestamp column of the row LCR does not match the timestamp of the corresponding table row. The row LCR is applied if its timestamp is later. Otherwise, the row LCR is discarded, and the table row is not changed. When you run the `ADD_AUTO_CDR` procedure, it adds an invisible timestamp column for each row in the specified table and configures timestamp conflict detection and resolution. When you use the `ADD_AUTO_CDR_COLUMN_GROUP` procedure to add one or more column groups, it adds a timestamp for the column group and configures timestamp conflict detection and resolution for the column group.

You can configure an Oracle GoldenGate administrator using the `GRANT_ADMIN_PRIVILEGE` procedure in the `DBMS_GOLDENGATE_ADM` package.

1. Connect to the inbound server database as a Oracle GoldenGate administrator.
2. Run the `ADD_AUTO_CDR` procedure and specify the table to configure for latest timestamp conflict detection and resolution.
3. Run the `ADD_AUTO_CDR_COLUMN_GROUP` procedure and specify one or more column groups in the table.
4. Repeat the previous steps in each Oracle Database that replicates the table.

Example 9-22 Configuring the Latest Timestamp Conflict Detection and Resolution for a Table

This example configures latest timestamp conflict detection and resolution for the `hr.employees` table.

```
BEGIN
  DBMS_GOLDENGATE_ADM.ADD_AUTO_CDR(
    SCHEMA_NAME => 'HR',
    TABLE_NAME => 'EMPLOYEES');
END;
/
```

Example 9-23 Configuring Column Groups

This example configures the following column groups for timestamp conflict resolution on the `HR.EMPLOYEES` table:

- The `JOB_IDENTIFIER_CG` column group includes the `JOB_ID`, `DEPARTMENT_ID`, and `MANAGER_ID` columns.
- The `COMPENSATION_CG` column group includes the `SALARY` and `COMMISSION_PCT` columns.

```
BEGIN
  DBMS_GOLDENGATE_ADM.ADD_AUTO_CDR_COLUMN_GROUP (
    SCHEMA_NAME      => 'HR',
    TABLE_NAME      => 'EMPLOYEES',
    COLUMN_LIST      => 'JOB_ID, DEPARTMENT_ID, MANAGER_ID',
    COLUMN_GROUP_NAME => 'JOB_IDENTIFIER_CG');
END;
/

BEGIN
  DBMS_GOLDENGATE_ADM.ADD_AUTO_CDR_COLUMN_GROUP (
    SCHEMA_NAME      => 'HR',
    TABLE_NAME      => 'EMPLOYEES',
    COLUMN_LIST      => 'SALARY, COMMISSION_PCT',
    COLUMN_GROUP_NAME => 'COMPENSATION_CG');
END;
/
```

Configuring Delta Conflict Detection and Resolution

The `ADD_AUTO_CDR_DELTA_RES` procedure in the `DBMS_GOLDENGATE_ADM` package configures delta conflict detection and resolution.

With delta conflict resolution, you specify one column for which conflicts are detected and resolved. The conflict is detected if the value of the column in the row LCR does not match the corresponding value in the table. The conflict is resolved by adding the difference between the new and old values in the row LCR to the value in the table.

You can configure an Oracle GoldenGate administrator using the `GRANT_ADMIN_PRIVILEGE` procedure in the `DBMS_GOLDENGATE_ADM` package.

1. Connect to the inbound server database as an Oracle GoldenGate administrator.
2. Run the `ADD_AUTO_CDR` procedure and specify the table to configure for latest timestamp conflict detection and resolution.
3. Run the `ADD_AUTO_CDR_DELTA_RES` procedure and specify the column on which delta conflict detection and resolution is performed.
4. Repeat the previous steps in each Oracle Database that replicates the table.

Example 9-24 Configuring Delta Conflict Detection and Resolution for a Table

This example configures delta conflict detection and resolution for the `order_total` column in the `oe.orders` table.

```
BEGIN
  DBMS_GOLDENGATE_ADM.ADD_AUTO_CDR (
    SCHEMA_NAME => 'OE',
    TABLE_NAME => 'ORDERS');
END;
/
```

```
BEGIN
  DBMS_GOLDENGATE_ADM.ADD_AUTO_CDR_DELTA_RES (
    SCHEMA_NAME => 'OE',
    TABLE_NAME => 'ORDERS',
    COLUMN_NAME => 'ORDER_TOTAL');
END;
/
```

Managing Automatic Conflict Detection and Resolution

You can manage Oracle GoldenGate automatic conflict detection and resolution in Oracle Database with the `DBMS_GOLDENGATE_ADM` package.

Altering Conflict Detection and Resolution for a Table

The `ALTER_AUTO_CDR` procedure in the `DBMS_GOLDENGATE_ADM` package alters conflict detection and resolution for a table.

Oracle GoldenGate automatic conflict detection and resolution must be configured for the table:

1. Connect to the inbound server database as the Oracle GoldenGate administrator.
2. Run the `ALTER_AUTO_CDR` procedure and specify the table to configure for latest timestamp conflict detection and resolution.
3. Repeat all of the previous steps in each Oracle Database that replicates the table.

Example 9-25 Altering Conflict Detection and Resolution for a Table

This example alters conflict detection and resolution for the `HR.EMPLOYEES` table to specify that delete conflicts are tracked in a tombstone table.

```
BEGIN
  DBMS_GOLDENGATE_ADM.ALTER_AUTO_CDR (
    SCHEMA_NAME      => 'HR',
    TABLE_NAME      => 'EMPLOYEES',
    TOMBSTONE_DELETES => TRUE);
END;
/
```

Altering a Column Group

The `ALTER_AUTO_CDR_COLUMN_GROUP` procedure alters a column group.

1. Connect to the inbound server database as an Oracle GoldenGate administrator.
2. Run the `ALTER_AUTO_CDR_COLUMN_GROUP` procedure and specify one or more column groups in the table.
3. Repeat all of the previous steps in each Oracle Database that replicates the table.

Example 9-26 Altering a Column Group

This example removes the `MANAGER_ID` column from the `JOB_IDENTIFIER_CG` column group for the `HR.EMPLOYEES` table.

```
BEGIN
  DBMS_GOLDENGATE_ADM.ALTER_AUTO_CDR_COLUMN_GROUP (
    SCHEMA_NAME      => 'HR',
    TABLE_NAME      => 'EMPLOYEES',
    COLUMN_GROUP_NAME => 'JOB_IDENTIFIER_CG',
    REMOVE_COLUMN_LIST => 'MANAGER_ID');
END;
/
```

**Note:**

If there is more than one column, then use a comma-separated list.

Purging Tombstone Rows

The `PURGE_TOMBSTONES` procedure removes tombstone rows that were recorded before a specified date and time. This procedure removes the tombstone rows for all tables configured for conflict resolution in the database.

It might be necessary to purge tombstone rows periodically to keep the tombstone log from growing too large over time.

1. Connect to the inbound server database as an Oracle GoldenGate administrator.
2. Run the `PURGE_TOMBSTONES` procedure and specify the date and time.

Example 9-27 Purging Tombstone Rows

This example purges all tombstone rows recorded before 3:00 p.m. on December, 1, 2015 Eastern Standard Time. The timestamp must be entered in `TIMESTAMP WITH TIME ZONE` format.

```
EXEC DBMS_GOLDENGATE_ADM.PURGE_TOMBSTONES('2015-12-01 15:00:00.000000 EST');
```

Removing Conflict Detection and Resolution From a Table

With Oracle Database 23ai and higher, removing Automatic Conflict Detection and Resolution (ACDR) entirely from the table has lesser impact on the table because the `AUTO_CDR`-related columns are marked as `UNUSED` if `AUTO_CDR` is removed.

After calling the `REMOVE_AUTO_CDR` procedure, the unused columns can be manually deleted in a maintenance window. This is useful for large tables where the `ALTER TABLE ... DROP COLUMN` operation is resource intensive.

If you want to remove all `AUTO_CDR` internal columns immediately when calling the `REMOVE_AUTO_CDR` procedure, you have to first mark the table using the `additional_options` parameter `REMOVE_HIDDEN_COLUMNS` for the `ADD_AUTO_CDR` or `ALTER_AUTO_CDR` procedure.

Use the `REMOVE_AUTO_CDR` procedure in the `DBMS_GOLDENGATE_ADM` package to tag a table as `UNUSED`, which minimizes blocking. You can choose to drop a column or retain it at a later stage.

1. Connect to the inbound server database as an Oracle GoldenGate administrator.
2. Run the `REMOVE_AUTO_CDR` procedure and specify the table.
3. Repeat all of the previous steps in each Oracle Database that replicates the table.

Example 9-28 Removing Conflict Detection and Resolution for a Table

This example removes conflict detection and resolution for the `HR.EMPLOYEES` table.

```
BEGIN
  DBMS_GOLDENGATE_ADM.REMOVE_AUTO_CDR(
    SCHEMA_NAME => 'HR',
    TABLE_NAME => 'EMPLOYEES');
END;
/
```

You can choose to drop columns by using the `ADD_AUTO_CDR.REMOVE_HIDDEN_COLUMNS` flag as an `additional_flags` parameter in the `ADD_AUTO_CDR` procedure.

Here is an example that you can use to view hidden columns in a table.

The following query uses the `DBA_UNUSED_COL_TABS` package to determine if there unused columns in the `EMPLOYEES` table.

```
SELECT OWNER, TABLE_NAME, COUNT
FROM   DBA_UNUSED_COL_TABS
WHERE  OWNER = 'HR'
AND    TABLE_NAME = 'EMPLOYEES'
ORDER BY OWNER, TABLE_NAME;
```

The output displays as follows:

OWNER	TABLE_NAME	COUNT
HR	EMPLOYEES	1

The following query lists out the hidden columns that were tagged by the system when ACDR was removed for the column group in the `EMPLOYEES` table.

```
SELECT OWNER, TABLE_NAME, COLUMN_ID, COLUMN_NAME, DATA_TYPE, HIDDEN_COLUMN
FROM   DBA_TAB_COLS
WHERE  OWNER = 'HR'
AND    TABLE_NAME = 'EMPLOYEES'
AND    HIDDEN_COLUMN = 'YES' AND USER_GENERATED= 'NO'
ORDER BY OWNER, TABLE_NAME, COLUMN_ID;
```

The output displays as follows:

OWNER	TABLE_NAME	COLUMN_ID	COLUMN_NAME	DATA_TYPE	HIDDEN_COLUMN
HR	EMPLOYEES	SYS_C00014_22092220:30:52\$		TIMESTAMP(6)	YES

Removing a Column Group

With Oracle Database 23ai and higher, removing Automatic Conflict Detection and Resolution (ACDR) from column groups has lesser impact on the table because the ACDR related columns are marked as `UNUSED`. You can also choose to drop a column or retain it at a later stage.

Use the `REMOVE_AUTO_CDR_COLUMN_GROUP` procedure in the `DBMS_GOLDENGATE_ADM` package to tag a table, which minimizes blocking. See the example in [Removing Conflict Detection and Resolution From a Table](#).

1. Connect to the inbound server database as an Oracle GoldenGate administrator.
2. Run the `REMOVE_AUTO_CDR_COLUMN_GROUP` procedure and specify the name of the column group.
3. Repeat all of the previous steps in each Oracle Database that replicates the table.

Example 9-29 Removing a Column Group

This example removes the `COMPENSATION_CG` column group from the `HR.EMPLOYEES` table.

```
BEGIN
  DBMS_GOLDENGATE_ADM.REMOVE_AUTO_CDR_COLUMN_GROUP (
    SCHEMA_NAME      => 'HR',
    TABLE_NAME      => 'EMPLOYEES',
    COLUMN_GROUP_NAME => 'COMPENSATION_CG');
END;
/
```

Removing Delta Conflict Detection and Resolution

The `REMOVE_AUTO_CDR_DELTA_RES` procedure in the `DBMS_GOLDENGATE_ADM` package removes delta conflict detection and resolution for a column.

Delta conflict detection and resolution must be configured for the specified column.

1. Connect to the inbound server database as an Oracle GoldenGate administrator.
2. Run the `REMOVE_AUTO_CDR_DELTA_RES` procedure and specify the column.
3. Repeat all of the previous steps in each Oracle Database that replicates the table.

Example 9-30 Removing Delta Conflict Detection and Resolution for a Table

This example removes delta conflict detection and resolution for the `ORDER_TOTAL` column in the `OE.ORDERS` table.

```
BEGIN
  DBMS_GOLDENGATE_ADM.REMOVE_AUTO_CDR_DELTA_RES (
    SCHEMA_NAME => 'OE',
    TABLE_NAME => 'ORDERS',
    COLUMN_NAME => 'ORDER_TOTAL');
END;
/
```

Monitoring Automatic Conflict Detection and Resolution

You can monitor Oracle GoldenGate automatic conflict detection and resolution in an Oracle Database by querying data dictionary views.

Displaying Information About the Tables Configured for Conflicts

The `ALL_GG_AUTO_CDR_TABLES` view displays information about the tables configured for Oracle GoldenGate automatic conflict detection and resolution.

1. Connect to the database.
2. Query the `ALL_GG_AUTO_CDR_TABLES` view.

Example 9-31 Displaying Information About the Tables Configured for Conflict Detection and Resolution

This query displays the following information about the tables that are configured for conflict detection and resolution:

- The table owner for each table.
- The table name for each table.
- The tombstone table used to store rows deleted for update-delete conflicts, if a tombstone table is configured for the table.
- The hidden timestamp column used for conflict resolution for each table.

```
COLUMN TABLE_OWNER FORMAT A15
COLUMN TABLE_NAME FORMAT A15
COLUMN TOMBSTONE_TABLE FORMAT A15
COLUMN ROW_RESOLUTION_COLUMN FORMAT A25
```

```
SELECT TABLE_OWNER,
       TABLE_NAME,
       TOMBSTONE_TABLE,
       ROW_RESOLUTION_COLUMN
FROM ALL_GG_AUTO_CDR_TABLES
ORDER BY TABLE_OWNER, TABLE_NAME;
```

Your output looks similar to the following:

TABLE_OWNER	TABLE_NAME	TOMBSTONE_TABLE	ROW_RESOLUTION_COLUMN
HR	EMPLOYEES	DT\$_EMPLOYEES	CDRTS\$ROW
OE	ORDERS	DT\$_ORDERS	CDRTS\$ROW

Displaying Information About Conflict Resolution Columns

The `ALL_GG_AUTO_CDR_COLUMNS` view displays information about the columns configured for Oracle GoldenGate automatic conflict detection and resolution.

The columns can be configured for row or column automatic conflict detection and resolution. The columns can be configured for latest timestamp conflict resolution in a column group. In addition, a column can be configured for delta conflict resolution.

1. Connect to the database as an Oracle GoldenGate administrator.
2. Query the `ALL_GG_AUTO_CDR_COLUMNS` view.

Example 9-32 Displaying Information About Column Groups

This query displays the following information about the tables that are configured for conflict detection and resolution:

- The table owner for each table.
- The table name for each table.
- If the column is in a column group, then the name of the column group.
- The column name.
- If the column is configured for latest timestamp conflict resolution, then the name of the hidden timestamp column for the column.

```
COLUMN TABLE_OWNER FORMAT A10
COLUMN TABLE_NAME FORMAT A10
COLUMN COLUMN_GROUP_NAME FORMAT A17
COLUMN COLUMN_NAME FORMAT A15
COLUMN RESOLUTION_COLUMN FORMAT A23
```

```
SELECT TABLE_OWNER,
       TABLE_NAME,
       COLUMN_GROUP_NAME,
       COLUMN_NAME,
       RESOLUTION_COLUMN
FROM ALL_GG_AUTO_CDR_COLUMNS
ORDER BY TABLE_OWNER, TABLE_NAME;
```

Your output looks similar to the following:

TABLE_OWNE	TABLE_NAME	COLUMN_GROUP_NAME	COLUMN_NAME	RESOLUTION_COLUMN
HR	EMPLOYEES	COMPENSATION_CG	COMMISSION_PCT	CDRTS\$COMPENSATION_CG
HR	EMPLOYEES	COMPENSATION_CG	SALARY	CDRTS\$COMPENSATION_CG
HR	EMPLOYEES	JOB_IDENTIFIER_CG	MANAGER_ID	
CDRTS\$JOB_IDENTIFIER_CG				
HR	EMPLOYEES	JOB_IDENTIFIER_CG	JOB_ID	
CDRTS\$JOB_IDENTIFIER_CG				
HR	EMPLOYEES	JOB_IDENTIFIER_CG	DEPARTMENT_ID	
CDRTS\$JOB_IDENTIFIER_CG				
HR	EMPLOYEES	IMPLICIT_COLUMNS\$	PHONE_NUMBER	CDRTS\$ROW
HR	EMPLOYEES	IMPLICIT_COLUMNS\$	LAST_NAME	CDRTS\$ROW
HR	EMPLOYEES	IMPLICIT_COLUMNS\$	HIRE_DATE	CDRTS\$ROW
HR	EMPLOYEES	IMPLICIT_COLUMNS\$	FIRST_NAME	CDRTS\$ROW
HR	EMPLOYEES	IMPLICIT_COLUMNS\$	EMAIL	CDRTS\$ROW
HR	EMPLOYEES	IMPLICIT_COLUMNS\$	EMPLOYEE_ID	CDRTS\$ROW
OE	ORDERS	IMPLICIT_COLUMNS\$	ORDER_MODE	CDRTS\$ROW
OE	ORDERS	IMPLICIT_COLUMNS\$	ORDER_ID	CDRTS\$ROW
OE	ORDERS	IMPLICIT_COLUMNS\$	ORDER_DATE	CDRTS\$ROW
OE	ORDERS	IMPLICIT_COLUMNS\$	CUSTOMER_ID	CDRTS\$ROW
OE	ORDERS	DELTA\$	ORDER_TOTAL	
OE	ORDERS	IMPLICIT_COLUMNS\$	PROMOTION_ID	CDRTS\$ROW
OE	ORDERS	IMPLICIT_COLUMNS\$	ORDER_STATUS	CDRTS\$ROW
OE	ORDERS	IMPLICIT_COLUMNS\$	SALES_REP_ID	CDRTS\$ROW

In this example, the columns with `IMPLICIT_COLUMNS$` for the column group name are configured for row conflict detection and resolution, but they are not part of a column group. The columns with `DELTA$` for the column group name are configured for delta conflict detection and resolution, and these columns do not have a resolution column.

Displaying Information About Column Groups

The `ALL_GG_AUTO_CDR_COLUMN_GROUPS` view displays information about the column groups configured for Oracle GoldenGate automatic conflict detection and resolution.

You can configure Oracle GoldenGate automatic conflict detection and resolution using the `ADD_AUTO_CDR` procedure in the `DBMS_GOLDENGATE_ADM` package. You can configure column groups using the `ADD_AUTO_CDR_COLUMN_GROUP` procedure in the `DBMS_GOLDENGATE_ADM` package.

1. Connect to the database as an Oracle GoldenGate administrator.
2. Query the `ALL_GG_AUTO_CDR_COLUMN_GROUPS` view.

Example 9-33 Displaying Information About Column Groups

This query displays the following information about the tables that are configured for conflict detection and resolution:

- The table owner.
- The table name.
- The name of the column group.
- The hidden timestamp column used for conflict resolution for each column group.

```
COLUMN TABLE_OWNER FORMAT A15
COLUMN TABLE_NAME FORMAT A15
COLUMN COLUMN_GROUP_NAME FORMAT A20
COLUMN RESOLUTION_COLUMN FORMAT A25

SELECT TABLE_OWNER,
       TABLE_NAME,
       COLUMN_GROUP_NAME,
       RESOLUTION_COLUMN
FROM ALL_GG_AUTO_CDR_COLUMN_GROUPS
ORDER BY TABLE_OWNER, TABLE_NAME;
```

The output looks similar to the following:

TABLE_OWNER	TABLE_NAME	COLUMN_GROUP_NAME	RESOLUTION_COLUMN
HR	EMPLOYEES	COMPENSATION_CG	CDRTS\$COMPENSATION_CG
HR	EMPLOYEES	JOB_IDENTIFIER_CG	CDRTS\$JOB_IDENTIFIER_CG

Manual Conflict Detection and Resolution

Learn about manually configuring Conflict Detection and Resolution (CDR) using specific parameters. Conflict detection and resolution is required in active-active configurations, where Oracle GoldenGate must maintain data synchronization among multiple databases that contain the same data sets.

Overview of the Oracle GoldenGate CDR Feature

Oracle GoldenGate Conflict Detection and Resolution (CDR) has two parts: Conflict Detection and Conflict Resolution. Before starting with conflict resolution, it's important to investigate and complete conflict detection.

Oracle GoldenGate Conflict Detection and Resolution (CDR) provides basic conflict resolution routines that:

- Resolve a uniqueness conflict for an `INSERT`.
- Resolve a "no data found" conflict for an `UPDATE` when the row exists, but the before image of one or more columns is different from the current value in the database.
- Resolve a "no data found" conflict for an `UPDATE` when the row does not exist.
- Resolve a "no data found" conflict for a `DELETE` when the row exists, but the before image of one or more columns is different from the current value in the database.
- Resolve a "no data found" conflict for a `DELETE` when the row does not exist.

To use conflict detection and resolution (CDR), the target database must reside on a Windows, Linux, or UNIX system. It is not supported for databases on the NonStop platform.

CDR supports scalar data types such as:

- `NUMERIC`
- `BOOLEAN`
- `DATE`
- `TIMESTAMP`
- `CHAR/NCHAR`
- `VARCHAR/ NVARCHAR`

This means that these column types can be used with the `COMPARECOLS` parameter and as the resolution column in the `USEMIN` and `USEMAX` options of the `RESOLVECONFLICT` parameter. Only `NUMERIC` columns can be used for the `USEDELTA` option of `RESOLVECONFLICT`. For `USEMAX`, `USEMIN`, only `TIMESTAMP` and `NUMBER` are supported.

Conflict resolution is not performed when Replicat operates in `BATCHSQL` mode. If a conflict occurs in `BATCHSQL` mode, Replicat reverts to `GROUPTRANSOPS` mode, and then to single-transaction mode. Conflict detection occurs in all three modes.

Configuring the Oracle GoldenGate Parameter Files for Error Handling

Manual CDR should be used in conjunction with error handling to capture errors that were resolved and errors that CDR could not resolve.

1. Conflict resolution is performed before these other error-handling parameters: `HANDLECATIONS`, `INSERTMISSINGUPDATES`, and `REPERROR`. Use the `REPERROR` parameter to assign rules for handling errors that cannot be resolved by CDR, or for errors that you do not want to handle through CDR. It might be appropriate to have `REPERROR` handle some errors, and CDR handle others; however, if `REPERROR` and CDR are configured to handle the same conflict, CDR takes precedence. The `INSERTMISSINGUPDATES` and `HANDLECATIONS` parameters also can be used to handle some errors not handled by CDR. See the *Parameters and Functions Reference for Oracle GoldenGate* for details about these parameters.

2. (Optional) Create an exceptions table. When an exceptions table is used with an exceptions `MAP` statement, Replicat sends every operation that generates a conflict (resolved or not) to the exceptions `MAP` statement to be mapped to the exceptions table. Omit a primary key on this table if Replicat is to process `UPDATE` and `DELETE` conflicts; otherwise there can be integrity constraint errors.

At minimum, an exceptions table should contain the same columns as the target table. These rows will contain each row image that Replicat applied to the target (or tried to apply).

In addition, you can define additional columns to capture other information that helps put the data in transactional context. Oracle GoldenGate provides tools to capture this information through the exceptions `MAP` statement. Such columns can be, but are not limited to, the following:

- The before image of the trail record. This is a duplicate set of the target columns with names such as `col1_before`, `col2_before`, and so forth.
 - The current values of the target columns. This also is a duplicate set of the target columns with names such as `col1_current`, `col2_current`, and so forth.
 - The name of the target table
 - The timestamp of the conflict
 - The operation type
 - The database error number
 - (Optional) The database error message
 - Whether the conflict was resolved or not
3. Create an exceptions `MAP` statement to map the exceptions data to the exceptions table. An exceptions `MAP` statement contains:
 - (Required) The `INSERTALLRECORDS` option. This parameter converts all mapped operations to `INSERTS` so that all column values are mapped to the exceptions table.
 - (Required) The `EXCEPTIONSONLY` option. This parameter causes Replicat to map operations that generate an error, but not those that were successful.
 - (Optional) A `COLMAP` clause. If the names and definitions of the columns in the exceptions table are identical to those of the source table, and the exceptions table only contains those columns, no `COLMAP` is needed. However, if any names or definitions differ, or if there are extra columns in the exceptions table that you want to populate with additional data, use a `COLMAP` clause to map all columns.

Tools for Mapping Extra Data to the Exceptions Table

The following are some tools that you can use in the `COLMAP` clause to populate extra columns:

- If the names and definitions of the source columns are identical to those of the target columns in the exceptions table, you can use the `USEDEFAULTS` keyword instead of explicitly mapping names. Otherwise, you must map those columns in the `COLMAP` clause, for example:

```
COLMAP (exceptions_col1 = col1, [...])
```

- To map the before image of the source row to columns in the exceptions table, use the `@BEFORE` conversion function, which captures the before image of a column from the trail record. This example shows the `@BEFORE` usage.

```
COLMAP (USEDEFAULTS, exceptions_col1 = @BEFORE (source_col1), &
exceptions_col2 = @BEFORE (source_col2), [...])
```

- To map the current image of the target row to columns in the exceptions table, use a `SQLEXEC` query to capture the image, and then map the results of the query to the columns in the exceptions table by using the '`queryID.column`' syntax in the `COLMAP` clause, as in the following example:

```
COLMAP (USEDEFAULTS, name_current = queryID.name, phone_current =
queryID.phone, [...])
```

- To map timestamps, database errors, and other environmental information, use the appropriate Oracle GoldenGate column-conversion functions. For example, the following maps the current timestamp at time of execution.

```
res_date = @DATENOW ()
```

See [Sample Exceptions Mapping with Additional Columns in the Exceptions Table](#) , for how to combine these features in a `COLMAP` clause in the exceptions `MAP` statement to populate a detailed exceptions table.

See Reference for Oracle GoldenGate for Windows and UNIX for the usage and syntax of the parameters and column-conversion functions shown in these examples.

Sample Exceptions Mapping with Source and Target Columns Only

The following is a sample parameter file that shows error handling and simple exceptions mapping for the source and target tables that are used in the CDR examples that begin. This example maps source and target columns, but no extra columns. For the following reasons, a `COLMAP` clause is not needed in the exceptions `MAP` statement in this example:

- The source and target exceptions columns are identical in name and definition.
- There are no other columns in the exceptions table.

Note:

This example intentionally leaves out other parameters that are required in a Replicat parameter file, such as process name and login credentials, as well as any optional parameters that may be required for a given database type. When using line breaks to split a parameter statement into multiple lines, use an ampersand (&) at the end of each line.

```
-- REPERERROR error handling: DEFAULT represents all error types. DISCARD
-- writes operations that could not be processed to a discard file.
REPERERROR (DEFAULT, DISCARD)
-- Specifies a discard file.
DISCARDFILE /users/ogg/discards/discards.dsc, PURGE
-- The regular MAP statement with the CDR parameters
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)), &
);
-- Starts the exceptions MAP statement by mapping the source table to the
-- exceptions table.
MAP fin.src, TARGET fin.exception, &
-- directs Replicat only to map operations that caused the error specified
-- in REPERROW.
EXCEPTIONSONLY, &
-- directs Replicat to convert all the exceptions to inserts into the
-- exceptions table. This is why there cannot be a primary key constraint
-- on the exceptions table.
INSERTALLRECORDS
;

```

Sample Exceptions Mapping with Additional Columns in the Exceptions Table

The following is a sample parameter file that shows error handling and complex exceptions mapping for the source and target tables that are used in the CDR examples that begin. In this example, the exceptions table has the same rows as the source table, but it also has additional columns to capture context data.



Note:

This example intentionally leaves out other parameters that are required in a Replicat parameter file, such as process name and login credentials, as well as any optional parameters that may be required for a given database type. When using line breaks to split a parameter statement into multiple lines, use an ampersand (&) at the end of each line.

```

-- REPERROW error handling: DEFAULT represents all error types. DISCARD
-- writes operations that could not be processed to a discard file.
REPERROW (DEFAULT, DISCARD)
-- Specifies the discard file.
DISCARDFILE /users/ogg/discards/discards.dsc, PURGE
-- The regular MAP statement with the CDR parameters
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))
);
-- Starts the exceptions MAP statement by mapping the source table to the
-- exceptions table.
MAP fin.src, TARGET fin.exception, &
-- directs Replicat only to map operations that caused the error specified
-- in REPERROW.
EXCEPTIONSONLY, &
-- directs Replicat to convert all the exceptions to inserts into the
-- exceptions table. This is why there cannot be a primary key constraint

```

```

-- on the exceptions table.
INSERTALLRECORDS &
-- SQLEXEC query to select the values from the target record before the
-- Replicat statement is applied. These are mapped to the *_target
-- columns later.
SQLEXEC (id qry, query 'select name, phone, address, salary, balance, &
comment, last_mod_time from fin.tgt where name = :p1', PARAMS(p1 = name)), &
-- Start of the column mapping, specifies use default column definitions.
COLMAP ( &
-- USEDEFAULTS maps the source columns to the target exceptions columns
-- that receive the after image that Replicat applied or tried to apply.
-- In this case, USEDEFAULTS can be used because the names and
definitions
-- of the source and target exceptions columns are identical; otherwise
-- the columns must be mapped explicitly in the COLMAP clause.
USEDEFAULTS, &
-- captures the timestamp when the resolution was performed.
res_date = @DATENOW (), &
-- captures and maps the DML operation type.
optype = @GETENV ('LASTERR', 'OPTYPE'), &
-- captures and maps the database error number that was returned.
dberrnum = @GETENV ('LASTERR', 'DBERRNUM'), &
-- captures and maps the database error that was returned.
dberrmsg = @GETENV ('LASTERR', 'DBERRMSG'), &
-- captures and maps the name of the target table
tablename = @GETENV ('GGHEADER', 'TABLENAME'), &
-- If the names and definitions of the source columns and the target
-- exceptions columns were not identical, the columns would need to
-- be mapped in the COLMAP clause instead of using USEDEFAULTS, as
-- follows:
-- name_after = name, &
-- phone_after = phone, &
-- address_after = address, &
-- salary_after = salary, &
-- balance_after = balance, &
-- comment_after = comment, &
-- last_mod_time_after = last_mod_time &
-- maps the before image of each column from the trail to a column in the
-- exceptions table.
name_before = @BEFORE (name), &
phone_before = @BEFORE (phone), &
address_before = @BEFORE (address), &
salary_before = @BEFORE (salary), &
balance_before = @BEFORE (balance), &
comment_before = @BEFORE (comment), &
last_mod_time_before = @BEFORE (last_mod_time), &
-- maps the results of the SQLEXEC query to rows in the exceptions table
-- to show the current image of the row in the target.
name_current = qry.name, &
phone_current = qry.phone, &
address_current = qry.address, &
salary_current = qry.salary, &
balance_current = qry.balance, &
comment_current = qry.comment, &
last_mod_time_current = qry.last_mod_time)
;

```

Once you are confident that your routines work as expected in all situations, you can reduce the amount of data that is logged to the exceptions table to reduce the overhead of the resolution routines.

Configuring the Oracle GoldenGate Parameter Files for Conflict Resolution

The following parameters are required to support conflict detection and resolution.

1. Use the `COMPARECOLS` option of the `MAP` parameter in the Replicat parameter file to specify columns that are to be used with before values in the Replicat `WHERE` clause. The before values are compared with the current values in the target database to detect update and delete conflicts. (By default, Replicat only uses the primary key in the `WHERE` clause; this may not be enough for conflict detection).
2. Use the `RESOLVECONFLICT` option of the `MAP` parameter to specify conflict resolution routines for different operations and conflict types. You can use `RESOLVECONFLICT` multiple times in a `MAP` statement to specify different resolutions for different conflict types. However, you cannot use `RESOLVECONFLICT` multiple times for the same type of conflict. Use identical conflict-resolution procedures on all databases, so that the same conflict produces the same end result. One conflict-resolution method might not work for every conflict that could occur. You might need to create several routines that can be called in a logical order of priority so that the risk of failure is minimized.



Note:

Additional consideration should be given when a table has a primary key and additional unique indexes or unique keys. The automated routines provided with the `COMPARECOLS` and `RESOLVECONFLICT` parameters require a consistent way to uniquely identify each row. Failure to consistently identify a row will result in an error during conflict resolution. In these situations the additional unique keys should be disabled or you can use the `SQLEXEC` feature to handle the error thrown and resolve the conflict.

For detailed information about these parameters, see *Parameters and Functions Reference for Oracle GoldenGate*. See the examples starting on [CDR Example 1: All Conflict Types with USEMAX, OVERWRITE, DISCARD](#), for more information on these parameters.

Making the Required Column Values Available to Extract

To use CDR, the following column values must be logged so that Extract can write them to the trail.

- The full before image of each record. Some databases do not provide a before image in the log record, and must be configured to do so with supplemental logging. For most supported databases, you can use the `ADD TRANDATA` command for this purpose.
- Use the `LOGALLSUPCOLS` parameter to ensure that the full before and after images of the scheduling columns are written to the trail. Scheduling columns are primary key, unique index, and foreign key columns. `LOGALLSUPCOLS` causes Extract to include in the trail record the before image for `UPDATE` operations and the before image of all supplementally logged columns for both `UPDATE` and `DELETE` operations.

For detailed information about these parameters and commands, see the *Parameters and Functions Reference for Oracle GoldenGate*. See the examples starting on [CDR Example 1:](#)

[All Conflict Types with USEMAX, OVERWRITE, DISCARD](#) for more information on how these parameters work with CDR.

Viewing CDR Statistics

The CDR feature provides the following methods for viewing the results of conflict resolution.

Here are different techniques you can use to view CDR statistics.

Report File

Replicat writes CDR statistics to the report file:

```
Total CDR conflicts          7
  CDR resolutions succeeded    6
  CDR resolutions failed      1
  CDR INSERTROWEXISTS conflicts 1
  CDR UPDATEROWEXISTS conflicts 4
  CDR UPDATEROWMISSING conflicts
  CDR DELETEROWEXISTS conflicts 1
  CDR DELETEROWMISSING conflicts 1
```

Command Line

You can view CDR statistics from the command line by using the `STATS REPLICAT` command with the `REPORTCDR` option:

```
STATS REPLICAT group, REPORTCDR
```

Column-conversion Functions

The following CDR statistics can be retrieved and mapped to an exceptions table or used in other Oracle GoldenGate parameters that accept input from column-conversion functions, as appropriate.

- Number of conflicts that Replicat detected
- Number of resolutions that the Replicat resolved
- Number of resolutions that the Replicat could not resolve

To retrieve these statistics, use the `@GETENV` column-conversion function with the `STATS` or `DELTASTATS` information type. The results are based on the current Replicat session. If Replicat stops and restarts, it resets the statistics.

You can return these statistics for a specific table or set of wildcarded tables:

```
@GETENV ('STATS', 'TABLE', 'SCHEMA.TABLNAME', 'CDR_CONFLICTS')
@GETENV ('STATS', 'TABLE', 'SCHEMA.TABLNAME', 'CDR_RESOLUTIONS_SUCCEEDED')
@GETENV ('STATS', 'TABLE', 'SCHEMA.TABLNAME', 'CDR_RESOLUTIONS_FAILED')
```

You can return these statistics for all of the tables in all of the `MAP` statements in the Replicat parameter file:

```
@GETENV ('STATS','CDR_CONFLICTS')
@GETENV ('STATS','CDR_RESOLUTIONS_SUCCEEDED')
@GETENV ('STATS','CDR_RESOLUTIONS_FAILED')
```

The 'STATS' information type in the preceding examples can be replaced by 'DELTASTATS' to return the requested counts since the last execution of 'DELTASTATS'. For more information about @GETENV, see @GETENV

CDR Example 1: All Conflict Types with USEMAX, OVERWRITE, DISCARD

This example resolves all conflict types by using the `USEMAX`, `OVERWRITE`, and `DISCARD` resolutions.

Table Used in this Example

The examples assume identical Oracle databases.

```
CREATE TABLE hr.emp_tgt(
  name varchar2(30) primary key,
  phone varchar2(10),
  address varchar2(100),
  salary number,
  balance number,
  comment varchar2(100),
  last_mod_time timestamp);
```

At the source database, all columns are supplementally logged:

```
ADD TRANDATA hr.emp_src, COLS (name, phone, address, salary, balance, comment,
last_mod_time);
```

MAP Statement with Conflict Resolution Specifications

```
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)),
  );
```

Description of MAP Statement

The following describes the `MAP` statement:

- Per `COMPARECOLS`, use the before image of all columns in the trail record in the Replicat `WHERE` clause for updates and deletes.
- Per `DEFAULT`, use all columns as the column group for all conflict types; thus the resolution applies to all columns.
- For an `INSERTROWEXISTS` conflict, use the `USEMAX` resolution: If the row exists during an insert, use the `last_mod_time` column as the resolution column for deciding which is the greater value: the value in the trail or the one in the database. If the value in the trail is

greater, apply the record but change the insert to an update. If the database value is higher, ignore the record.

- For an `UPDATEROWEXISTS` conflict, use the `USEMAX` resolution: If the row exists during an update, use the `last_mod_time` column as the resolution column: If the value in the trail is greater, apply the update.
- If you use `USEMIN` or `USEMAX`, and the values are exactly the same, then `RESOLVECONFLICT` isn't triggered and the incoming row is ignored. If you use `USEMINEQ` or `USEMAXEQ`, and the values are exactly the same, then the resolution is triggered.
- For a `DELETEROWEXISTS` conflict, use the `OVERWRITE` resolution: If the row exists during a delete operation, apply the delete.
- For an `UPDATEROWMISSING` conflict, use the `OVERWRITE` resolution: If the row does not exist during an update, change the update to an insert and apply it.
- For a `DELETROWMISSING` conflict use the `DISCARD` resolution: If the row does not exist during a delete operation, discard the trail record.

Note:

As an alternative to `USEMAX`, you can use the `USEMAXEQ` resolution to apply a `>=` condition. For more information, see *Parameters and Functions Reference for Oracle GoldenGate*.

INSERTROWEXISTS with the USEMAX Resolution

For this example, the `USEMAX` resolution is illustrated with the applicable before and after images for the record in the trail and in the database. It shows how to resolve an insert where the row exists in the source and target, but some or all row values are different.

Table 9-9 INSERTROWEXISTS Conflict with USEMAX Resolution

Image	SQL	Comments
Before image in trail	None (row was inserted on the source).	N/A
After image in trail	<pre>name='Mary' phone='1234567890' address='Oracle Pkwy' salary=100 balance=100 comment=NULL last_mod_time='9/1/10 3:00'</pre>	<code>last_mod_time='9/1/10 3:00'</code> is the after image of the resolution column. Since there is an after image, this will be used to determine the resolution.
Target database image	<pre>name='Mary' phone='111111' address='Ralston' salary=200 balance=500 comment='aaa' last_mod_time='9/1/10 1:00'</pre>	<code>last_mod_time='9/1/10 1:00'</code> is the current image of the resolution column in the target against which the resolution column value in the trail is compared.

Table 9-9 (Cont.) INSERTROWEXISTS Conflict with USEMAX Resolution

Image	SQL	Comments
Initial INSERT applied by Replicat that detects the conflict	SQL bind variables: 1) 'Mary' 2) '1234567890' 3) 'Oracle Pkwy' 4) 100 5) 100 6) NULL 7) '9/1/10 3:00'	This SQL returns a uniqueness conflict on 'Mary'.
UPDATE applied by Replicat to resolve the conflict	SQL bind variables: 1) '1234567890' 2) 'Oracle Pkwy' 3) 100 4) 100 5) NULL 6) '9/1/10 3:00' 7) 'Mary' 8) '9/1/10 3:00'	Because USEMAX is specified for INSERTROWEXISTS, Replicat converts the insert to an update, and it compares the value of last_mod_time in the trail record with the value in the database. The value in the record is greater, so the after images for columns in the trail file are applied to the target.

UPDATEROWEXISTS with the USEMAX Resolution

For this example, the USEMAX resolution is illustrated with the applicable before and after images for the record in the trail and in the database. It shows how to resolve an update where the row exists in the source and target, but some or all row values are different.

Table 9-10 UPDATEROWEXISTS Conflict with USEMAX Resolution

Image	SQL	Comments
Before image in trail	name='Mary' phone='1234567890' address='Oracle Pkwy' salary=100 balance=100 comment=NULL last_mod_time='9/1/10 3:00'	last_mod_time='9/1/10 3:00 is the before image of the resolution column.
After image in trail	phone='222222' address='Holly' last_mod_time='9/1/10 5:00'	last_mod_time='9/1/10 5:00 is the after image of the resolution column. Since there is an after image, this will be used to determine the resolution.
Target database image	name='Mary' phone='1234567890' address='Oracle Pkwy' salary=100 balance=600 comment='com' last_mod_time='9/1/10 6:00'	last_mod_time='9/1/10 6:00 is the current image of the resolution column in the target against which the resolution column value in the trail is compared.

Table 9-10 (Cont.) UPDATEROWEXISTS Conflict with USEMAX Resolution

Image	SQL	Comments
Initial UPDATE applied by Replicat that detects the conflict	SQL bind variables: 1) '222222' 2) 'Holly' 3) '9/1/10 5:00' 4) 'Mary' 5) '1234567890' 6) 'Oracle Pkwy' 7) 100 8) 100 9) NULL 10) '9/1/10 3:00'	This SQL returns a no-data-found error because the values for the balance, comment, and last_mod_time are different in the target. All columns are used in the WHERE clause because the COMPARECOLS statement is set to ALL.
UPDATE applied by Replicat to resolve the conflict	SQL bind variables: 1) 'Mary' 2) '222222' 3) 'Holly' 4) 100 5) 100 6) NULL 7) '9/1/10 5:00' 8) 'Mary' 9) '9/1/10 5:00'	Because the after value of last_mod_time in the trail record is less than the current value in the database, the database value is retained. Replicat applies the operation with a WHERE clause that contains the primary key plus a last_mod_time value set to less than 9/1/10 5:00. No rows match this criteria, so the statement fails with a "data not found" error, but Replicat ignores the error because a USEMAX resolution is expected to fail if the condition is not satisfied.

UPDATEROWMISSING with OVERWRITE Resolution

For this example, the **OVERWRITE** resolution is illustrated with the applicable before and after images for the record in the trail and in the database. It shows how to resolve the case where the target row is missing. The logical resolution, and the one used, is to overwrite the row into the target so that both databases are in sync again.

Table 9-11 UPDATEROWMISSING Conflict with OVERWRITE Resolution

Image	SQL	Comments
Before image in trail	name='Jane' phone='333' address='Oracle Pkwy' salary=200 balance=200 comment=NULL last_mod_time='9/1/10 7:00'	N/A
After image in trail	phone='4444' address='Holly' last_mod_time='9/1/10 8:00'	
Target database image	None (row for Jane is missing)	

Table 9-11 (Cont.) UPDATEROWMISSING Conflict with OVERWRITE Resolution

Image	SQL	Comments
Initial UPDATE applied by Replicat that detects the conflict	SQL bind variables: 1) '4444' 2) 'Holly' 3) '9/1/10 8:00' 4) 'Jane' 5) '333' 6) 'Oracle Pkwy' 7) 200 8) 200 9) NULL 10) '9/1/10 7:00'	This SQL returns a no-data-found error. All columns are used in the WHERE clause because the COMPARECOLS statement is set to ALL.
INSERT applied by Replicat to resolve the conflict	SQL bind variables: 1) 'Jane' 2) '4444' 3) 'Holly' 4) 200 5) 200 6) NULL 7) '9/1/10 8:00'	The update is converted to an insert because OVERWRITE is the resolution. The after image of a column is used if available; otherwise the before image is used.

DELETEROWEXISTS with OVERWRITE Resolution

For this example, the OVERWRITE resolution is illustrated with the applicable before and after images for the record in the trail and in the database. It shows how to resolve the case where the source row was deleted but the target row exists. In this case, the OVERWRITE resolution applies the delete to the target.

Table 9-12 DELETEROWEXISTS Conflict with OVERWRITE Resolution

Image	SQL	Comments
Before image in trail	name='Mary' phone='222222' address='Holly' salary=100 balance=100 comment=NULL last_mod_time='9/1/10 5:00'	N/A
After image in trail	None	N/A
Target database image	name='Mary' phone='1234567890' address='Oracle Pkwy' salary=100 balance=600 comment=com last_mod_time='9/1/10 7:00'	The row exists on the target, but the phone, address, balance, comment, and last_mod_time columns are different from the before image in the trail.

Table 9-12 (Cont.) DELETETEROWEXISTS Conflict with OVERWRITE Resolution

Image	SQL	Comments
Initial DELETE applied by Replicat that detects the conflict	SQL bind variables: 1) 'Mary' 2) '222222' 3) 'Holly' 4) 100 5) 100d 6) NULL 7) '9/1/10 5:00'	All columns are used in the WHERE clause because the COMPARECOLS statement is set to ALL. A no-data-found error occurs because of the difference between the before and current values.
DELETE applied by Replicat to resolve the conflict	SQL bind variables: 1) 'Mary'	Because OVERWRITE is the resolution, the DELETE is applied using only the primary key (to avoid an integrity error).

DELETETEROWMISSING with DISCARD Resolution

For this example, the DISCARD resolution is illustrated with the applicable before and after images for the record in the trail and in the database. It shows how to resolve the case where the target row is missing. In the case of a delete on the source, it is acceptable for the target row not to exist (it would need to be deleted anyway), so the resolution is to discard the DELETE operation that is in the trail.

Table 9-13 DELETETEROWMSING Conflict with DISCARD Resolution

Image	SQL	Comments
Before image in trail	name='Jane' phone='4444' address='Holly' salary=200 balance=200 comment=NULL last_mod_time='9/1/10 8:00'	N/A
After image in trail	None	N/A
Target database image	None (row missing)	N/A
Initial DELETE applied by Replicat that detects the conflict	SQL bind variables: 1) 'Jane' 2) '4444' 3) 'Holly' 4) 200 5) 200 6) NULL 7) '9/1/10 8:00'	This SQL returns a no-data-found error. All columns are used in the WHERE clause because the COMPARECOLS statement is set to ALL.
SQL applied by Replicat to resolve the conflict	None	Because DISCARD is specified as the resolution for DELETETEROWMISSING, so the delete from the trail goes to the discard file.

CDR Example 2: UPDATEROWEXISTS with USEDELTA and USEMAX

This example resolves the condition where a target row exists on `UPDATE` but non-key columns are different, and it uses two different resolution types to handle this condition based on the affected column.

Table Used in this Example

The examples assume identical Oracle databases.

```
CREATE TABLE tgt(  
    name varchar2(30) primary key,  
    phone varchar2(10),  
    address varchar2(100),  
    salary number,  
    balance number,  
    comment varchar2(100),  
    last_mod_time timestamp);
```

At the source database, all columns are supplementally logged:

```
ADD TRANDATA scott.src, COLS (name, phone, address, salary, balance, comment,  
    last_mod_time);
```

MAP Statement

```
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)));
```

Description of MAP Statement

For an `UPDATEROWEXISTS` conflict, where a target row exists on `UPDATE` but non-key columns are different, use two different resolutions depending on the column:

- Per the `delta_res_method` resolution, use the `USEDELTA` resolution logic for the `salary` column so that the change in value will be added to the current value of the column.
- Per `DEFAULT`, use the `USEMAX` resolution logic for all other columns in the table (the default column group), using the `last_mod_time` column as the resolution column. This column is updated with the current time whenever the row is modified; the value of this column in the trail is compared to the value in the target. If the value of `last_mod_time` in the trail record is greater than the current value of `last_mod_time` in the target database, the changes to `name`, `phone`, `address`, `balance`, `comment` and `last_mod_time` are applied to the target.

Per `COMPARECOLS`, use the primary key (`name` column) plus the `address`, `phone`, `salary`, and `last_mod_time` columns as the comparison columns for conflict detection for `UPDATE` and `DELETE` operations. (The `balance` and `comment` columns are not compared.)

**Note:**

As an alternative to USEMAX, you can use the USEMAXEQ resolution to apply a >= condition. For more information, see *Parameters and Functions Reference for Oracle GoldenGate*.

Error Handling

For an example of error handling to an exceptions table, see [Configuring the Oracle GoldenGate Parameter Files for Error Handling](#).

Table 9-14 UPDATEROWEXISTS with USEDELTA and USEMAX

Image	SQL	Comments
Before image in trail	<pre>name='Mary' phone='1234567890' address='Oracle Pkwy' salary=100 balance=100 comment=NULL last_mod_time='9/1/10 3:00'</pre>	<p>last_mod_time='9/1/10 3:00' is the before image of the resolution column for the USEMAX resolution.</p> <p>salary=100 is the before image for the USEDELTA resolution.</p>
After image in trail	<pre>phone='222222' address='Holly' salary=200 comment='new' last_mod_time='9/1/10 5:00'</pre>	<p>last_mod_time='9/1/10 5:00' is the after image of the resolution column for USEMAX. Since there is an after image, this will be used to determine the resolution.</p>
Target database image	<pre>name='Mary' phone='1234567890' address='Oracle Pkwy' salary=600 balance=600 comment='com' last_mod_time='9/1/10 4:00'</pre>	<p>last_mod_time='9/1/10 4:00' is the current image of the resolution column in the target against which the resolution column value in the trail is compared.</p> <p>salary=600 is the current image of the target column for the USEDELTA resolution.</p>

Table 9-14 (Cont.) UPDATEROWEXISTS with USEDELTA and USEMAX

Image	SQL	Comments
Initial UPDATE applied by Replicat that detects the conflict	SQL bind variables: 1) '222222' 2) 'Holly' 3) 200 4) 'new' 5) '9/1/10 5:00' 6) 'Mary' 7) '1234567890' 8) 'Oracle Pkwy' 9) 100 10) '9/1/10 3:00'	This SQL returns a no-data-found error because the values for the <code>salary</code> and <code>last_mod_time</code> are different. (The values for <code>comment</code> and <code>balance</code> are also different, but these columns are not compared.)
UPDATE applied by Replicat to resolve the conflict for <code>salary</code> , using <code>USEDELTA</code> .	SQL bind variables: 1) 200 2) 100 3) 'Mary'	Per <code>USEDELTA</code> , the difference between the after image of <code>salary</code> (200) in the trail and the before image of <code>salary</code> (100) in the trail is added to the current value of <code>salary</code> in the target (600). The result is 700. $600 + (200 - 100) = 700$
UPDATE applied by Replicat to resolve the conflict for the default columns, using <code>USEMAX</code> .	SQL bind variables: 1) '222222' 2) 'Holly' 3) 'new' 4) '9/1/10 5:00' 5) 'Mary' 6) '9/1/10 5:00'	Per <code>USEMAX</code> , because the after value of <code>last_mod_time</code> in the trail record is greater than the current value in the database, the row is updated with the after values from the trail record. Note that the <code>salary</code> column is not set here, because it is resolved with the UPDATE from the <code>USEDELTA</code> resolution.

CDR Example 3: UPDATEROWEXISTS with USEDELTA, USEMAX, and IGNORE

This example resolves the conflict where a target row exists on `UPDATE` but non-key columns are different, and it uses three different resolution types to handle this condition based on the affected column.

Table Used in this Example

The examples assume identical Oracle databases.

```
CREATE TABLE tgt(
  name varchar2(30) primary key,
  phone varchar2(10),
  address varchar2(100),
  salary number,
  balance number,
  comment varchar2(100),
  last_mod_time timestamp);
```

At the source database, all columns are supplementally logged:

```
ADD TRANDATA scott.src, COLS (name, phone, address, salary, balance, comment,  
last_mod_time);
```

MAP Statement

```
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));
```

Description of MAP Statement

- For an `UPDATEROWEXISTS` conflict, where a target row exists on `UPDATE` but non-key columns are different, use two different resolutions depending on the column:
 - Per the `delta_res_method` resolution, use the `USEDELTA` resolution logic for the `salary` and `balance` columns so that the change in each value will be added to the current value of each column.
 - Per the `max_res_method` resolution, use the `USEMAX` resolution logic for the `address` and `last_mod_time` columns. The `last_mod_time` column is the resolution column. This column is updated with the current time whenever the row is modified; the value of this column in the trail is compared to the value in the target. If the value of `last_mod_time` in the trail record is greater than the current value of `last_mod_time` in the target database, the changes to `address` and `last_mod_time` are applied to the target; otherwise, they are ignored in favor of the target values.
 - Per `DEFAULT`, use the `IGNORE` resolution logic for the remaining columns (`phone` and `comment`) in the table (the default column group). Changes to these columns will always be ignored by Replicat.
- Per `COMPARECOLS`, use all columns except the `comment` column as the comparison columns for conflict detection for `UPDATE` operations. `Comment` will not be used in the `WHERE` clause for updates, but all other columns that have a before image in the trail record will be used.

Note:

As an alternative to `USEMAX`, you can use the `USEMAXEQ` resolution to apply a `>=` condition. For more information, see *Parameters and Functions Reference for Oracle GoldenGate*.

Error Handling

For an example of error handling to an exceptions table, see [Configuring the Oracle GoldenGate Parameter Files for Error Handling](#).

Table 9-15 UPDATEROWEXISTS with USEDELTA, USEMAX, and IGNORE

Image	SQL	Comments
Before image in trail	<pre> name='Mary' phone='1234567890' address='Oracle Pkwy' salary=100 balance=100 comment=NULL last_mod_time='9/1/10 3:00 </pre>	<p>last_mod_time='9/1/10 3:00 is the before image of the resolution column for the USEMAX resolution.</p> <p>salary=100 and balance=100 are the before images for the USEDELTA resolution.</p>
After image in trail	<pre> phone='222222' address='Holly' salary=200 comment='new' last_mod_time='9/1/10 5:00' </pre>	<p>last_mod_time='9/1/10 5:00 is the after image of the resolution column for USEMAX. Since there is an after image, this will be used to determine the resolution.</p> <p>salary=200 is the only after image available for the USEDELTA resolution. For balance, the before image will be used in the calculation.</p>
Target database image	<pre> name='Mary' phone='1234567890' address='Ralston' salary=600 balance=600 comment='com' last_mod_time='9/1/10 4:00' </pre>	<p>last_mod_time='9/1/10 4:00 is the current image of the resolution column in the target against which the resolution column value in the trail is compared for USEMAX.</p> <p>salary=600 and balance=600 are the current images of the target columns for USEDELTA.</p>
Initial UPDATE applied by Replicat that detects the conflict	<p>SQL bind variables:</p> <pre> 1) '222222' 2) 'Holly' 3) 200 4) 'new' 5) '9/1/10 5:00' 6) 'Mary' 7) '1234567890' 8) 'Oracle Pkwy' 9) 100 10) 100 11) '9/1/10 3:00' </pre>	<p>This SQL returns a no-data-found error because the values for the address, salary, balance and last_mod_time columns are different.</p>
UPDATE applied by Replicat to resolve the conflict for salary, using USEDELTA.	<p>SQL bind variables:</p> <pre> 1) 200 2) 100 3) 'Mary' </pre>	<p>For salary, there is a difference of 100, but there was no change in value for balance, so it is not needed in the update SQL. Per USEDELTA, the difference (delta) between the after (200) image and the before image (100) of salary in the trail is added to the current value of salary in the target (600). The result is 700.</p>

Table 9-15 (Cont.) UPDATEROWEXISTS with USEDELTA, USEMAX, and IGNORE

Image	SQL	Comments
UPDATE applied by Replicat to resolve the conflict for USEMAX.	SQL bind variables: 1) 'Holly' 2) '9/1/10 5:00' 3) 'Mary' 4) '9/1/10 5:00'	Because the after value of <code>last_mod_time</code> in the trail record is greater than the current value in the database, that column plus the <code>address</code> column are updated with the after values from the trail record. Note that the <code>salary</code> column is not set here, because it is resolved with the UPDATE from the USEDELTA resolution.
UPDATE applied by Replicat for IGNORE.	SQL bind variables: 1) '222222' 2) 'new' 3) 'Mary'	IGNORE is specified for the DEFAULT column group (<code>phone</code> and <code>comment</code>), so no resolution SQL is applied.

Error Handling

Topics:

Overview of Oracle GoldenGate Error Handling

Oracle GoldenGate provides error-handling options for:

- Extract
- Replicat
- TCP/IP

Handling Extract Errors

There is no specific parameter to handle Extract errors when DML operations are being extracted, but Extract does provide a number of parameters that can be used to prevent anticipated problems. These parameters handle anomalies that can occur during the processing of DML operations, such as what to do when a row to be fetched cannot be located, or what to do when the transaction log is not available. The following is a partial list of these parameters.

- FETCHOPTIONS
- WARNLONGTRANS
- DBOPTIONS
- TRANLOGOPTIONS

To handle extraction errors that relate to DDL operations, use the `DDLERROR` parameter.

For a complete parameter list, see *Parameters and Functions Reference for Oracle GoldenGate*.

Handling Replicat Errors during DML Operations

To control the way that Replicat responds to an error during one of its DML statements, use the `REPERROR` parameter in the Replicat parameter file. You can use `REPERROR` as a global parameter or as part of a `MAP` statement. You can handle most errors in a default fashion (for example, to cease processing) with `DEFAULT` and `DEFAULT2` options, and also handle other errors in a specific manner.

The following comprise the range of `REPERROR` responses:

- `ABEND`: roll back the transaction and stop processing.
- `DISCARD`: log the error to the discard file and continue processing.
- `EXCEPTION`: send the error for exceptions processing.
- `IGNORE`: ignore the error and continue processing.
- `RETRYOP [MAXRETRIES n]`: retry the operation, optionally up to a specific number of times.
- `TRANSABORT [, MAXRETRIES n] [, DELAY[C]SECS n]`: abort the transaction and reposition to the beginning, optionally up to a specific number of times at specific intervals.
- `RESET`: remove all previous `REPERROR` rules and restore the default of `ABEND`.
- `TRANSDISCARD`: discard the entire replicated source transaction if any operation within that transaction, including the commit, causes a Replicat error that is listed in the error specification. This option is useful when integrity constraint checking is disabled on the target.
- `TRANSEXCEPTION`: perform exceptions mapping for every record in the replicated source transaction, according to its exceptions-mapping statement, if any operation within that transaction (including the commit) causes a Replicat error that is listed in the error specification.

Most options operate on the individual record that generated an error, and Replicat processes the other, successful operations in the transaction. The exceptions are `TRANSDISCARD` and `TRANSEXCEPTION`: These options affect all records in a transaction if any record in that transaction generates an error. (The `ABEND` option also applies to the entire transaction, but does not apply error handling.)

See `REPERROR` for syntax and usage.

Handling Errors as Exceptions

When the action of `REPERROR` is `EXCEPTION` or `TRANSEXCEPTION`, you can map the values of operations that generate errors to an exceptions table and, optionally, map other information about the error that can be used to resolve the error. See [About the Exceptions Table](#).

To map the exceptions to the exceptions table, use either of the following options of the `MAP` parameter:

- `MAP with EXCEPTIONSONLY`
- `MAP with MAPEXCEPTION`

Using `EXCEPTIONSONLY`

`EXCEPTIONSONLY` is valid for one pair of source and target tables that are explicitly named and mapped one-to-one in a `MAP` statement; that is, there cannot be wildcards. To use

EXCEPTIONSONLY, create two MAP statements for each source table that you want to use EXCEPTIONSONLY for on the target:

- The first, a standard MAP statement, maps the source table to the actual target table.
- The second, an *exceptions MAP statement*, maps the source table to the *exceptions table* (instead of to the target table). An exceptions MAP statement executes immediately after an error on the source table to send the row values to the exceptions table.

To identify a MAP statement as an exceptions MAP statement, use the INSERTALLRECORDS and EXCEPTIONSONLY options. The exceptions MAP statement must immediately follow the regular MAP statement that contains the same source table. Use a COLMAP clause in the exceptions MAP statement if the source and exceptions-table columns are not identical, or if you want to map additional information to extra columns in the exceptions table, such as information that is captured by means of column-conversion functions or SQLEXEC.

For more information about these parameters, see *Parameters and Functions Reference for Oracle GoldenGate*.

- A regular MAP statement that maps the source table `ggs.equip_account` to its target table `equip_account2`.
- An exceptions MAP statement that maps the same source table to the exceptions table `ggs.equip_account_exception`.

In this case, four extra columns were created, in addition to the same columns that the table itself contains:

```
DML_DATE
OPTYPE
DBERRNUM
DBERRMSG
```

To populate the `DML_DATE` column, the `@DATENOW` column-conversion function is used to get the date and time of the failed operation, and the result is mapped to the column. To populate the other extra columns, the `@GETENV` function is used to return the operation type, database error number, and database error message.

The `EXCEPTIONSONLY` option of the exceptions MAP statement causes the statement to execute only after a failed operation on the source table. It prevents every operation from being logged to the exceptions table.

The `INSERTALLRECORDS` parameter causes all failed operations for the specified source table, no matter what the operation type, to be logged to the exceptions table as *inserts*.



Note:

There can be no primary key or unique index restrictions on the exception table. Uniqueness violations are possible in this scenario and would generate errors.

Example 9-34 EXCEPTIONSONLY

This example shows how to use `REPERROR` with `EXCEPTIONSONLY` and an exceptions MAP statement. This example only shows the parameters that relate to `REPERROR`; other parameters not related to error handling are also required for Replicat.

```
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'));
```

In this example, the `REPERROR` parameter is set for `DEFAULT` error handling, and the `EXCEPTION` option causes the Replicat process to treat failed operations as exceptions and continue processing.

Using MAPEXCEPTION

`MAPEXCEPTION` is valid when the names of the source and target tables in the `MAP` statement are wildcarded. Place the `MAPEXCEPTION` clause in the regular `MAP` statement, the same one where you map the source tables to the target tables. Replicat maps all operations that generate errors from all of the wildcarded tables to the same exceptions table; therefore, the exceptions table should contain a superset of all of the columns in all of the wildcarded tables.

Because you cannot individually map columns in a wildcard configuration, use the `COLMAP` clause with the `USEDEFAULTS` option to handle the column mapping for the wildcarded tables (or use the `COLMATCH` parameter if appropriate), and use explicit column mappings to map any additional information, such as that captured with column-conversion functions or `SQLEXEC`.

When using `MAPEXCEPTION`, include the `INSERTALLRECORDS` parameter in the `MAPEXCEPTION` clause. `INSERTALLRECORDS` causes all operation types to be applied to the exceptions table as `INSERT` operations. This is required to keep an accurate record of the exceptions and to prevent integrity errors on the exceptions table.

For more information about these parameters, see *Parameters and Functions Reference for Oracle GoldenGate*.

Example 9-35 MAPEXCEPTION

This is an example of how to use `MAPEXCEPTION` for exceptions mapping. The `MAP` and `TARGET` clauses contain wildcarded source and target table names. Exceptions that occur when processing any table with a name beginning with `TRX` are captured to the `fin.trxexceptions` table using the designated 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')
)
);
```

About the Exceptions Table

Use an exceptions table to capture information about an error that can be used for such purposes as troubleshooting your applications or configuring them to handle the error. At minimum, an exceptions table should contain enough columns to receive the entire row image from the failed operation. You can define extra columns to contain other information that is captured by means of column-conversion functions, `SQLEXEC`, or other external means.

To ensure that the trail record contains values for all of the columns that you map to the exceptions table, you can use either the `LOGALLSUPCOLS` parameter or the following parameters in the Extract parameter file:

- Use the `NOCOMPRESSDELETES` parameter so that all columns of a row are written to the trail for `DELETE` operations.
- Use the `GETUPDATEBEFORES` parameter so that Extract captures the before image of a row and writes them to the trail.

Handling Replicat Errors during DDL Operations

To control the way that Replicat responds to an error that occurs for a DDL operation on the target, use the `DDLERROR` parameter in the Replicat parameter file.

For more information, see `DDLERROR`.

Handling TCP/IP Errors

To provide instructions for responding to TCP/IP errors, use the `TCPERRS` file. This file is in the Oracle GoldenGate directory

Table 9-16 TCPERRS Columns

Column	Description
Error	Specifies a TCP/IP error for which you are defining a response.
Response	Controls whether or not Oracle GoldenGate tries to connect again after the defined error. Valid values are either <code>RETRY</code> or <code>ABEND</code> .
Delay	Controls how long Oracle GoldenGate waits before attempting to connect again.
Max Retries	Controls the number of times that Oracle GoldenGate attempts to connect again before aborting.

If a response is not explicitly defined in the `TCPERRS` file, Oracle GoldenGate responds to TCP/IP errors by abending.

Example 9-36 TCPERRS File

```
# TCP/IP error handling parameters
# Default error response is abend
#
# Error          Response    Delay(csecs)  Max Retries
ECONNABORTED    RETRY          1000          10
ECONNREFUSED    RETRY          1000          12
ECONNRESET      RETRY          500           10
ENETDOWN        RETRY          3000          50
ENETRESET       RETRY          1000          10
ENOBUFS         RETRY          100           60
ENOTCONN        RETRY          100           10
EPIPE           RETRY          500           10
ESHUTDOWN       RETRY          1000          10
ETIMEDOUT       RETRY          1000          10
NODYNPORTS      RETRY          100           10
```

The `TCPERRS` file contains default responses to basic errors. To alter the instructions or add instructions for new errors, open the file in a text editor and change any of the values in the columns shown in [Table 9-16](#).

Maintaining Updated Error Messages

The error, information, and warning messages that Oracle GoldenGate processes generate are stored in a data file named `ggmessage.dat` in the Oracle GoldenGate installation directory. The version of this file is checked upon process startup and must be identical to that of the process in order for the process to operate.

Resolving Oracle GoldenGate Errors

To get help with specific troubleshooting issues, go to My Oracle Support at <https://support.oracle.com> and search the Knowledge Base.

Trail File Management

The Extract process captures the changes from the transaction logs of the source system (database) into trail files that are consumed by other Oracle GoldenGate processes.

Extract can write into one or multiple sets of trail files. A trail is a sequence of files that are created and aged as needed. Processes that read a trail are:

- **Replicat:** Replicat reads the trail file received on the target deployment.
- **Distribution Service:** Extracts data from a local trail for further processing, if needed, and transfers it to the target system.
- **Receiver Service:** Receives the trail and transfers to Replicat, which reads the trail and applies change data to the target database.

You can create more than one trail to separate the data of different tables or applications, or to satisfy the requirements of a specific replication topology, such as a cascading topology. You link tables specified with a `TABLE` statement to a trail specified with an `EXTTRAIL` parameter statement in the Extract parameter file.

- Assign Storage for Oracle GoldenGate Trails
- Estimate Space for the Trail
- Add a Trail

Also see [Using the LogDump Utility to Access Trail File Records](#).

Manage Trail Files

Assign Storage for Oracle GoldenGate Trails

In a typical configuration, there is at least one trail on the source system and one on the target system. Allocate enough disk space to allow for the following:

- The primary Extract process captures transactional data from the source database and writes it to the local trail. There must be enough disk space to contain the data accumulation, or the primary Extract will abend.

- For a trail at the target location, provide enough disk space to handle data accumulation according to the purge rules set with the `PURGEOLDEXTRACTS` parameter. Even with `PURGEOLDEXTRACTS` in use, data will always accumulate on the target because it is transferred across the network faster than it can be applied to the target database.

To prevent trail activity from interfering with business applications, assign a separate disk or file system to contain the trail files. Trail files can reside on drives that are local to the Oracle GoldenGate installation, or they can reside on NAS or SAN devices. In an Oracle cluster, they can reside on ASM or DBFS storage.

See [Preparing DBFS for an Active-Active Configuration](#).

Estimate Space for the Trails

The following are guidelines for estimating the amount of disk space that will be required to store Oracle GoldenGate trail data.

1. Estimate the longest time that the network could be unavailable. Plan to store enough data to withstand the longest possible outage, because otherwise you will need to resynchronize the source and target data if the outage outlasts disk capacity.
2. Estimate how much transaction log volume your business applications generate in one hour.
3. Use the following formula to calculate the required disk space.

```
[source transaction log volume in one hour] x [number of hours downtime] x .4  
= trail disk space
```

This equation uses a multiplier of 40 percent because only about 40 percent of the data in a transaction log is needed by Oracle GoldenGate.

Note:

This formula is a conservative estimate, and you should run tests once you have configured Oracle GoldenGate to determine exactly how much space you need. As a general observation, in case of subset replication, the required trail disk space might be much lower.

To prevent trail activity from interfering with business applications, assign a separate disk or file system to contain the trail files. Trail files can reside on local storage, network-attached storage (NAS, SAN), shared filesystem, or cluster file system (ACFS, DBFS).

Add a Trail

When you create, or add, a trail, you do not physically create any files on disk. The files are created automatically by an Extract process. Rather, you specify the name of the trail and associate it with the Extract group that writes to it.

You can add a trail, while you add an Extract from the Administration Service.

To add a trail from the command line interface, issue the following command on the source system:

```
ADD {EXTTRAIL} pathname, EXTRACT group [, MEGABYTES n]
```

This syntax includes:

- **EXTTRAIL:** This parameter specifies a trail on the local system.
- **pathname:** This option is the relative or fully qualified name of the trail, including a two-character name that can be any two alphanumeric characters, for example `c:\ggs\ea`. Oracle GoldenGate appends a serial number to each trail file as it is created during processing.
- **EXTRACT:** This option is the group name of the Extract that writes to this trail. Only one Extract group can write to a trail.
- **MEGABYTES n:** This is an optional argument with which you can set the size, in megabytes, of each trail file (default is 2000).

Example: Create a Local Trail

This example creates a local trail named `/ggs/ea` for Extract group `exte`.

```
ADD EXTTRAIL /ggs/ea, EXTRACT exte
```

You can also create a local trail using REST API. For more information, see [Create Trail](#).

Automate Maintenance Tasks

Use the **Tasks** tab on the Configuration page, to set up the following automated tasks.

Purging Trails

The Purge Trail page works the same way as the Manager `PURGEOLDEXTRACTS` parameter in the Classic Architecture. It allows you to purge trail files when Oracle GoldenGate has finished processing them. Automating this task ensures that the trail files are periodically deleted to avoid excessive consumption of disk space.

From the Tasks tab, when you select the Purge Trail page, it allows you to configure the Administration Service purge trail process.

1. Add a Purge Trail task by clicking the + sign .
2. Enter the **Operation Name** of the Administration Service task. The operation name is case sensitive. For example, you can create an operation with the name **TASK1** and another operation named **task1**.
3. Enter the trail path or trail name in the **Trail** field. The default trail file location is `$deployment_home/lib/data`.

If non-default trail file locations are used for storing and retrieving trail files, then the full path along with the trail name must be added. An example of the full path of the trail file location would be: `$deployment-home/lib/data/rep01/trail/et`.
4. Click the + sign to add the trail to the **Selected Trails** list.
5. If you don't need to use checkpoints, disable the option **Use Checkpoints**. However, Oracle recommends using checkpoints. If you don't use checkpoints, the trail will be purged whether or not it has been consumed if the keep rule is met.
6. Set the **Keep Rule** value to specify the maximum number of hours, days, or number of files for which the Purge Trails task needs to be active.
7. Specify the number of hours or days when the purge trails task has to run, in the Purge Frequency field and click Submit.
8. Use the Purge Trails task table to edit or delete the task, as required.

Also see PURGE EXTTRAIL.

Purging Tasks

You can automatically purge processes associated with an Administration Service.

From the Tasks tab, click Purge Tasks.

1. Enter the **Operation Name** that you need to set up for automatic purging.
2. Select the Extract or Replicat task (initial load process) **Process Name** for the operation. The list contains all processes so ensure that you select the correct task.
3. Select the Extract or Replicat task (initial load) **Process Type** for the operation.
4. If you enable **Use Stop Status**, the status of the task is used to perform the purge task.
5. Enter the hours or days after which you need to purge the process and click **Submit**.
6. Edit or delete the purge process task using the relevant icon from the Purge Tasks table.

Reporting Lag

You can manage lag reports from the Lag Report tab. To do so:

1. From the Tasks tab, click Lag Report.
2. The Action column contains all the options to delete, alter, refresh, and view the lag report task details.
3. Select the required option.
4. If you select the Alter Task option, you are presented with options to edit the lag report. The options are:
 - Enabled: To keep processing the lag report task.
 - Check Every (in minutes): To set a time interval to check the lag report.
 - Report: To log report for the task.
 - If Exceeds: To specify a threshold after which a warning would be initiated.
 - Warning: To allow a warning to be generated incase the lag threshold exceeds the specified limit.
 - When Exceeds: The lag threshold after which the warning is triggered.
5. Click Submit.

Admin Client Command Line Interface for Oracle GoldenGate Microservices

To start the Admin Client, you need to change the current working directory to the Oracle GoldenGate home directory (OGG_HOME).

For a complete list and description of commands available from the Admin Client, see About the Command Line Interfaces in the *Command Line Interface Reference for Oracle GoldenGate*.

About Admin Client

Admin Client is a command line utility. It uses the REST API published by the microservices to accomplish control and configuration tasks in an Oracle GoldenGate deployment.

Admin Client is a command line utility that can be used to create, modify, and remove Oracle GoldenGate processes and can be used in place of the MA web user interface. The Admin Client program is located in the `$OGG_HOME/bin` directory, where `$OGG_HOME` is the Oracle GoldenGate home directory.

If you need to automate the Admin Client connection with the deployment, you can use an Oracle Wallet to store the user credentials. The credentials stored must have the following characteristics:

- Single user name (account) and password
- Local to the environment where the Admin Client runs
- Available only to the currently logged user
- Managed by the Admin Client
- Referenced using a credential name
- Available for Oracle GoldenGate deployments and proxy connections.

To use the Admin Client for administration tasks, you need the user credentials that work with both the Service Manager and Administration Service. Here are the configurations required for working with the Admin Client:

1. Make sure that the `bin` directory of the Oracle Software is part of the `PATH` environment variable:

```
export PATH=ogg_install_location/bin:$PATH
```

If you configure a secure deployment using SSL certificate files (`.pem` or `.der`), you must add the `OGG_CLIENT_TLS_CAPATH` environment variable. This is required to be able to connect to the deployment from Admin Client. This variable is used to specify the location where the certificate files are located on the host. For clients only needing to validate server certificates, the `OGG_CLIENT_TLS_CAPATH` environment variable should refer to a file containing a trusted CA Certificate that is shared with the server to which the client is expected to connect.

```
export OGG_CLIENT_TLS_CAPATH = deployment_rootCA_certificate_location
```

**Note:**

For Microsoft Windows, the default certificate file format is `.der` while all other platforms use `.pem` as the default format.

2. Run the command:

```
[oracle]$ adminclient
```

The output displays the Oracle GoldenGate Admin Client prompt, where you can connect to the deployment from the Admin Client:

```
OGG (not connected) 1>
```

3. Connect to a deployment or to a proxy server from the Admin Client as a security user. This is the user you created while adding the deployment for your Oracle GoldenGate instance using OGGCA.

```
CONNECT http(s)://localhost:port DEPLOYMENT deployment name AS security
role user PASSWORD password
```

 **Note:**

If the password to connect to a secure or non-secure deployment from the Admin Client has an exclamation mark (!) at the end, then you must enter the password in double quotes when using the `CONNECT` command in a single line. Otherwise, the password is not accepted and the connection fails. This is required for all deployments with a strong password policy.

Syntax:

```
CONNECT server-url [ DEPLOYMENT deployment-name ]
[ ( ( AS deployment-credentials-name
  | USER deployment-user-name )
  [ PASSWORD deployment-password ] )
| TOKEN [ access-token ] ]
[ PROXY proxy-uri
[ ( AS proxy-credentials-name
  | USER proxy-user-name )
[ PASSWORD proxy-password ] ] ] [ ! ]
```

See the `CONNECT` command in the *Command Line Interface Reference for Oracle GoldenGate* to know more.

 **Note:**

The deployment credentials cannot be stored as a `USERIDALIAS` in the credential store because the Oracle wallet used for storing database credentials is managed by the Administration Service. Instead, a separate Oracle wallet is created for the Admin Client. The Oracle wallet is stored in the users home directory.

The following example shows adding an Oracle GoldenGate deployment user to connect to the deployment from the Admin Client:

```
ADD CREDENTIALS admin USER ggadmin PASSWORD *****
```

The password for the user is stored in the hidden GoldenGate directory `$HOME`.

Output:

```
2019-02-14T00:35:38Z INFO OGG-15114 Credential store altered.
```

The following example shows adding Oracle GoldenGate deployment proxy user to connect to the deployment from the Admin Client:

```
ADD CREDENTIALS proxy USER proxyadmin PASSWORD *****
```

Here's an example of using the `CONNECT` command to access a deployment using the Admin Client:

```
OGG (Not Connected)4> CONNECT http://www.example.com:12000 deployment EAST
PROXY http:111.1.1.1:3128 as proxyadmin password oggadmin-A2
Using default deployment 'Local'
OGG (http://www.example.com:12000 Local) 4>
```

4. You can view the full list of Admin Client commands using the `HELP` command. Use the `HELP SHOWSYNTAX` command to view the syntax for specific commands.

Using Wildcards in Command Arguments

You can use wildcards with certain Oracle GoldenGate commands to control multiple Extract and Replicat groups as a unit. The wildcard symbol that is supported by Oracle GoldenGate is the asterisk (*). An asterisk represents any number of characters. For example, to start all Extract groups whose names contain the letter X, issue the following command.

```
START EXTRACT *X*
```

Using Command History

The execution of multiple commands is made easier with the following tools:

- Use the `HISTORY` command to display a list of previously executed commands.
- Use the `!` command to execute a previous command again without editing it.
- Use the `FC` command to edit a previous command and then execute it again.

Storing and Calling Frequently Used Command Sequences

You can automate a frequently-used series of commands by using an `OBEY` file and the `OBEY` command. The `OBEY` file takes the character set of the local operating system. To specify a character that is not compatible with that character set, use the Unicode notation.

To use OBEY

1. Create and save a text file that contains the commands, one command per line. This is your `OBEY` file. The name can be anything supported by the operating system. You can nest other `OBEY` files within an `OBEY` file.
2. Run the Admin Client.
3. (Optional) If using an `OBEY` file that contains nested `OBEY` files, issue the following command. This command enables the use of nested `OBEY` files for the current session and is required whenever using nested `OBEY` files.

```
ALLOWNESTED
```

4. Call the OBEY file by using the OBEY command from the Admin Client.

```
OBEY file_name
```

Where:

file_name is the relative or fully qualified name of the OBEY file.

Example 9-37 OBEY command file

```
ALTER CREDENTIALSTORE ADD USER c##ggadmin@cdbl ALIAS cggwest DOMAIN
OracleGoldenGate PASSWORD ggadmin
DBLOGIN USERIDALIAS cggwest
ALTER CREDENTIALSTORE ADD USER ggadmin@pdbwest ALIAS ggwest DOMAIN
OracleGoldenGate PASSWORD Welcome2OGG

ADD SCHEMATRANDATA hr
ADD TRANDATA hr.employees
ADD HEARTBEATTABLE

ADD EXTRACT exte, INTEGRATED TRANLOG, BEGIN NOW
ADD EXTTRAIL east/ea, EXTRACT exte
START EXTRACT exte

INFO EXTRACT exte, DETAIL
```

See OBEY for more information in *Parameters and Functions Reference for Oracle GoldenGate*.

Controlling Extract and Replicat

Here are basic directions for controlling Extract and Replicat processes.

To Start Extract or Replicat

```
START {EXTRACT | REPLICAT} group_name
```

Where:

group_name is the name of the Extract or Replicat group or a wildcard set of groups (for example, * or fin*).

To Stop Extract or Replicat Gracefully

```
STOP {EXTRACT | REPLICAT} group_name
```

Where:

group_name is the name of the Extract or Replicat group or a wildcard set of groups (for example, * or fin*).

To Stop Replicat Forcefully

```
STOP REPLICAT group_name !
```

The current transaction is aborted and the process stops immediately. You cannot stop Extract forcefully.

To End a Process that STOP Cannot Stop

```
KILL {EXTRACT | REPLICAT} group_name
```

Ending a process does not shut it down gracefully, and checkpoint information can be lost.

To Control Multiple Processes at Once

```
command ER wildcard specification
```

Where:

- *command* can be KILL, START, or STOP
- *wildcard specification* is a wildcard specification for the names of the process groups that you want to affect with the command. The command affects every Extract and Replicat group that satisfies the wildcard. Oracle GoldenGate supports up to 100,000 wildcard entries.

Deleting Extract and Replicat

This section contains basic directions for deleting Extract and Replicat processes.

To Delete an Extract Group

1. Connect to the deployment from the Admin Client.
2. Issue the `DBLOGIN` command as the Extract database user (or a user with the same privileges). You can use either of the following commands, depending on whether a local credential store exists.

```
DBLOGIN [SOURCEDB dsn] {USERID user, PASSWORD password  
[encryption_options] | USERIDALIAS alias [DOMAIN domain]}
```

3. Stop the Extract process.

```
STOP EXTRACT group_name
```

4. Issue the following command.

```
DELETE EXTRACT group_name
```

5. (Oracle) Unregister the Extract group from the database.

```
UNREGISTER EXTRACT group_name,database_name
```

To Delete a Replicat Group

1. Stop the Replicat process.

```
STOP REPLICAT group_name
```

2. Issue one of the following commands to log into the database.

```
DBLOGIN [SOURCEDB dsn] {USERID user, PASSWORD password
[encryption_options] | USERIDALIAS alias [DOMAIN domain]}
```

Where:

- *SOURCEDB dsn* supplies the data source name, if required as part of the connection information.
- *USERID user*, *PASSWORD password* specifies an explicit database login credential.
- *USERIDALIAS alias* [*DOMAIN domain*] specifies an alias and optional domain of a credential that is stored in a local credential store.
- *encryption_options* is one of the options that encrypt the password.

3. Issue the following command to delete the group.

```
DELETE REPLICAT group_name
```

Deleting a Replicat group preserves the checkpoints in the checkpoint table (if being used). Deleting a process group also preserves the parameter file. You can create the same group again, using the same parameter file, or you can delete the parameter file to remove the group's configuration permanently.

Specifying Object Names in Oracle GoldenGate Input

The following rules apply when specifying object names in parameter files (such as in `TABLE` and `MAP` statements), column-conversion functions, commands, and in other input.

Specifying Filesystem Path Names in Parameter Files on Windows Systems

On Windows systems, if the name of any directory in a filesystem path name begins with a number, the path must be specified with forward slashes, not backward slashes, when listing that path in Oracle GoldenGate input, such as parameter files or commands. This requirement prevents Oracle GoldenGate from interpreting the name as an octal escape sequence. For example, the following paths contain a directory named `\2023` that will be interpreted as the octal sequence `\202`:

```
C:\deployments\ea
C:\deployments\north\ea
C:\deployments\north\2023\ea
```

The preceding path can be used with forward slashes as follows:

```
C:/deployments/ea
C:/deployments/north/ea
```

For more information, see [Support for Escape Sequences](#).

Supported Database Object Names

Object names in parameter files, command, and other input can be any length and in any supported character set. For supported character sets, see [Supported Character Sets](#).

Oracle GoldenGate supports most characters in object and column names. Specify object names in double quote marks if they contain special characters such as white spaces or symbols.

The following lists of supported and non-supported characters covers all databases supported by Oracle GoldenGate; a given database platform may or may not support all listed characters.

Supported Special Characters

Oracle GoldenGate supports all characters that are supported by the database, including the following special characters. Object names that contain these special characters must be enclosed within double quotes in parameter files.

Character	Description
/	Forward slash (See Specifying Names that Contain Slashes)
*	Asterisk (Must be escaped by a backward slash when used in parameter file, as in: *)
?	Question mark (Must be escaped by a backward slash when used in parameter file, as in: \?)
@	At symbol (Supported, but is often used as a resource locator by databases. May cause problems in object names)
#	Pound symbol
\$	Dollar symbol
%	Percent symbol (Must be %% when used in parameter file)
^	Caret symbol
()	Open and close parentheses
_	Underscore
-	Dash
<space>	Space

Non-supported Special Characters

The following characters are not supported in object names and non-key column names.

Character	Description
\	Backward slash (Must be \\ when used in parameter file)
{ }	Begin and end curly brackets (braces)
[]	Begin and end brackets
=	Equal symbol
+	Plus sign
!	Exclamation point
~	Tilde
	Pipe
&	Ampersand
:	Colon
;	Semi-colon

Character	Description
,	Comma
' '	Single quotes
" "	Double quotes
'	Accent mark (Diacritical mark)
.	Period
<	Less-than symbol (or beginning angle bracket)
>	Greater-than symbol (or ending angle bracket)

Specifying Names that Contain Slashes

If a table name contains a forward-slash character (/) in any part of its name, that name component must be enclosed within double quotes unless the object name is from an IBM i platform. The following are some examples:

```
"c/d"  
"/a".b  
a."b/"
```

If the name contains a forward slash that is not enclosed within double quotes, Oracle GoldenGate treats it as a name that originated on the IBM i platform (from a DB2 for i database). The forward slash in the name is interpreted as a separator character.

Qualifying Database Object Names

Object names must be fully qualified in the parameter file. This means that every name specification must be qualified, not only those supplied as input to Oracle GoldenGate parameter syntax, but also names in a SQL procedure or query that is supplied as `SQLEXEC` input, names in user exit input, and all other input supplied in the parameter file.

Oracle GoldenGate supports two-part and three-part object names, as appropriate for the database.

Two-part Names

Most databases require only two-part names to be specified, in the following format:

owner.object

For example: `HR.EMP`

Where:

owner is a schema or database, depending on how the database defines a logical namespace that contains database objects. *object* is a table or other supported database object.

The databases for which Oracle GoldenGate supports two-part names are as follows, shown with their appropriate two-part naming convention:

- Db2 for i: *schema.object* and *library/file(member)*
- Db2 LUW: *schema.object*
- Db2 on z/OS: *schema.object*

- MySQL: *database.object*
- Oracle Database (non-CDB databases): *schema.object*
- SQL Server: *schema.object*
- Teradata: *database.object*

Three-part Names

Oracle GoldenGate supports three-part names for Oracle container database, only in case of using downstream Extract. However, Oracle GoldenGate supports only per-PDB Extract for Oracle database, in general.

Three-part names are required to capture from a source Oracle container database because one Extract group can capture from more than one container. Thus, the name of the container, as well as the schema, must be specified for each object or objects in an Extract `TABLE` statement.

Specify a three-part Oracle CDB name as follows:

container.schema.object

For example: `PDBEAST.HR.EMP`

Applying Data from Multiple Containers or Catalogs

To apply data captured from multiple source containers or catalogs to a target Oracle container database, both three- and two-part names are required. In the `MAP` portion of the `MAP` statement, each source object must be associated with a container or catalog, just as it was in the `TABLE` statement. This enables you (and Replicat) to properly map data from multiple source containers or catalogs to the appropriate target objects. In the `TARGET` portion of the `MAP` statement, however, only two-part names are required. This is because Replicat can connect to only one target container or catalog at a time, and *schema.owner* is a sufficient qualifier. Multiple Replicat groups are required to support multiple target containers or catalogs. Specify the target container or catalog with the `TARGETTDB` parameter.

Specifying a Default Container or Catalog

You can use the `SOURCECATALOG` parameter to specify a default catalog for any subsequent `TABLE`, `MAP`, (or Oracle `SEQUENCE`) specifications in the parameter file.

The following example shows the use of `SOURCECATALOG` to specify the default Oracle PDB named `pdbeast` for `region` and `jobs` objects, and the default PDB named `pdwest` for `appraisal` objects. The objects in `pdbeast` are specified with a fully qualified three-part name, which does not require a default catalog to be specified.

```
TABLE pdbeast.hr.emp*;  
SOURCECATALOG pdbeast  
TABLE region.country*;  
TABLE jobs.desg*;  
SOURCECATALOG pdwest  
TABLE appraisal.sal*;
```

Specifying Case-Sensitive Database Object Names

Oracle GoldenGate supports case-sensitive names. Follow these rules when specifying case-sensitive objects.

- Specify object names from a case-sensitive database in the same case that is used to store them in the host database. Keep in mind that, in some database types, different levels of the database can have different case-sensitivity, such as case-sensitive schema but case-insensitive table. If the database requires quotes to enforce case-sensitivity, put quotes around each object that is case-sensitive in the qualified name.

Correct: `TABLE "Sales"."ACCOUNT"`

Incorrect: `TABLE "Sales.ACCOUNT"`

- Oracle GoldenGate converts case-insensitive names to the case in which they are stored when required for mapping purposes.

[Table 9-17](#) provides an overview of the support for case-sensitivity in object names, per supported database. Refer to the database documentation for details on this type of support.

Table 9-17 Case Sensitivity of Object Names Per Database

Database	Requires quotes to enforce case-sensitivity?	Unquoted object name	Quoted object name
DB2	Yes. Differentiates between case-sensitive and case-insensitive by use of quotes.	Case-insensitive, stores in upper case	Case-sensitive, stores in mixed case
MySQL (Case-sensitive database)	No <ul style="list-style-type: none"> Always case-sensitive, stores in mixed case The names of columns, triggers, and procedures are case-insensitive 	No effect	No effect
Oracle Database	Yes. Differentiates between case-sensitive and case-insensitive by use of quotes.	Case-insensitive, stores in upper case	Case-sensitive, stores in mixed case
SQL Server (Database created as case-sensitive)	No Always case-sensitive, stores in mixed case	No effect	No effect
SQL Server (Database created as case-insensitive)	No Always case-insensitive, stores in mixed case	No effect	No effect
Teradata	No Always case-insensitive, stores in mixed case	No effect	No effect

**Note:**

For all supported databases, passwords are always treated as case-sensitive regardless of whether the associated object name is quoted or unquoted.

Using Wildcards in Database Object Names

You can use wildcards for any part of a fully qualified object name, if supported for the specific database. These name parts can be the following: the container, database, or catalog name, the owner (schema or database name), and table or sequence name. For specifics on how object names and wildcards are supported, see the Oracle GoldenGate installation and configuration guide for that database.

Where appropriate, Oracle GoldenGate parameters permit the use of two wildcard types to specify multiple objects in one statement:

- A question mark (?) replaces one character. For example in a schema that contains tables named `TAB n` , where n is from 0 to 9, a wildcard specification of `HQ.TAB?` returns `HQ.TAB0`, `HQ.TAB1`, `HQ.TAB2`, and so on, up to `HQ.TAB9`, but no others. This wildcard is not supported for the DB2 LUW database nor for DEFGN. This wildcard can only be used to specify source objects in a `TABLE` or `MAP` parameter. It cannot be used to specify target objects in the `TARGET` clause of `TABLE` or `MAP`.
- An asterisk (*) represents any number of characters (including zero sequence). For example, the specification of `HQ.T*` could return such objects as `HQ.TOTAL`, `HQ.T123`, and `HQ.T`. This wildcard is valid for all database types throughout all Oracle GoldenGate commands and parameters where a wildcard is allowed.
- In `TABLE` and `MAP` statements, you can combine the asterisk and question-mark wildcard characters in source object names only.

Rules for Using Wildcards for Source Objects

For source objects, you can use the asterisk alone or with a partial name. For example, the following source specifications are valid:

- `TABLE HQ.*;`
- `TABLE PDB*.HQ.*;`
- `MAP HQ.T_*;`
- `MAP HQ.T_*, TARGET HQ.*;`

The `TABLE`, `MAP` and `SEQUENCE` parameters take the case-sensitivity and locale of the database into account for wildcard resolution. For databases that are created as case-sensitive or case-insensitive, the wildcard matches the exact name and case. For example, if the database is case-sensitive, `SCHEMA.TABLE` is matched to `SCHEMA.TABLE`, `Schema.Table` is matched to `Schema.Table`, and so forth. If the database is case-insensitive, the matching is not case-sensitive.

For databases that can have both case-sensitive and case-insensitive object names in the same database instance, with the use of quote marks to enforce case-sensitivity, the wildcarding works differently. When used alone for a source name in a `TABLE` statement, an asterisk wildcard matches any character, whether or not the asterisk is within quotes. The following statements produce the same results:

```
TABLE hr.*;
TABLE hr."*";
```

Similarly, a question mark wildcard used alone matches any single character, whether or not it is within quotes. The following produce the same results:

```
TABLE hr.?;
TABLE hr."?";
```

If a question mark or asterisk wildcard is used with other characters, case-sensitivity is applied to the non-wildcard characters, but the wildcard matches both case-sensitive and case-insensitive names.

- The following `TABLE` statements capture any table name that begins with lower-case `abc`. The quoted name case is preserved and a case-sensitive match is applied. It captures table names that include `"abcA"` and `"abca"` because the wildcard matches both case-sensitive and case-insensitive characters.

```
TABLE hr."abc*";
TABLE hr."abc?";
```

- The following `TABLE` statements capture any table name that begins with upper-case `ABC`, because the partial name is case-insensitive (no quotes) and is stored in upper case by this database. However, because the wildcard matches both case-sensitive and case-insensitive characters, this example captures table names that include `ABCA` and `"ABCa"`.

```
TABLE hr.abc*;
TABLE hr.abc?;
```

Rules for Using Wildcards for Target Objects

When using wildcards in the `TARGET` clause of a `MAP` statement, the target objects must exist in the target database. (The exception is when DDL replication is being used, which allows new schemas and their objects to be replicated as they are created.)

For target objects, only an asterisk can be used. If an asterisk wildcard is used with a partial name, Replicat replaces the wildcard with the entire name of the corresponding source object. Therefore, specifications such as the following are *incorrect*:

```
TABLE HQ.T_*, TARGET RPT.T_*;
MAP HQ.T_*, TARGET RPT.T_*;
```

The preceding mappings produce incorrect results, because the wildcard in the target specification is replaced with `T_TEST` (the name of a source object), making the whole target name `T_T_TESTn`. The following illustrates the incorrect results:

- `HQ.T_TEST1` maps to `RPT.T_T_TEST1`
- `HQ.T_TEST2` maps to `RPT.T_T_TEST2`
- (The same pattern applies to all other `HQ.T_TESTn` mappings.)

The following examples show the correct use of asterisk wildcards.

```
MAP HQ.T_*, TARGET RPT.*;
```

The preceding example produces the following correct results:

- `HQ.T_TEST1` maps to `RPT.T_TEST1`
- `HQ.T_TEST2` maps to `RPT.T_TEST2`
- (The same pattern applies to all other `HQ.T_TESTn` mappings.)

Fallback Name Mapping

Oracle GoldenGate has a fallback mapping mechanism in the event that a source name cannot be mapped to a target name. If an exact match cannot be found on the target for a case-sensitive source object, Replicat tries to map the source name to the same name in upper or lower case (depending on the database type) on the target. Fallback name mapping is controlled by the `NAMEMATCH` parameters. For more information, see *Parameters and Functions Reference for Oracle GoldenGate*.

Asterisks or Question Marks as Literals in Object Names

If the name of an object itself includes an asterisk or a question mark, the entire name must be escaped and placed within double quotes, as in the following example:

```
TABLE HT."\"?ABC";
```

How Wildcards are Resolved

By default, when an object name is wildcarded, the resolution for that object occurs when the first row from the source object is processed. (By contrast, when the name of an object is stated explicitly, its resolution occurs at process startup.) To change the rules for resolving wildcards, use the `WILDCARDRESOLVE` parameter. The default is `DYNAMIC`.

Excluding Objects from a Wildcard Specification

You can combine the use of wildcard object selection with explicit object exclusion by using the `EXCLUDEWILDCARDOBJECTSONLY`, `CATALOGEXCLUDE`, `SCHEMAEXCLUDE`, `MAPEXCLUDE`, and `TABLEEXCLUDE` parameters.

Differentiating Case-Sensitive Column Names from Literals

By default, Oracle GoldenGate follows SQL-92 rules for specifying column names and literals. In Oracle GoldenGate parameter files, conversion functions, user exits, and commands, case-sensitive column names must be enclosed within double quotes if the database requires quotes around a name to support case-sensitivity. For example:

```
"columnA"
```

Case-sensitive column names in databases that do not require quotes to enforce case-sensitivity must be specified as they are stored in the database. For example:

```
ColumnA
```

Literals must be enclosed within single quotes. In the following example, `Product_Code` is a case-sensitive column name in an Oracle database, and the other strings are literals.

```
@CASE ("Product_Code", 'CAR', 'A car', 'TRUCK', 'A truck')
```

Creating a Parameter File Using Admin Client

To create a parameter file, run the `EDIT PARAMS` command from the Admin Client. When you create a parameter file with `EDIT PARAMS`, 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`, but you also must specify the full path name with the `PARAMS` option of the `ADD EXTRACT` or `ADD REPLICAT` command when you create your process groups. After pairing with an Extract or Replicat group, a parameter file

must remain in its original location for Oracle GoldenGate to operate properly after processing has started.

The `EDIT PARAMS` command launches the following text editors in Admin Client:

- Notepad on Microsoft Windows systems.
- The vi editor on UNIX and Linux systems. Db2 for i only supports vi when connected with SSH or xterm. For more information, see [Creating a Parameter File with a Text Editor](#).



Note:

You can change the default editor through Admin Client by using the `SET EDITOR` command.

1. Run the Admin Client.
2. Connect to the Admin Client using the `CONNECT` command.
3. In Admin Client, issue the following command to open the default text editor:

```
EDIT PARAMS group_name
```

In this code snippet:

group_name is the name of the Extract or Replicat group for which the file is being created. The name of an Extract or Replicat parameter file must match that of the process group.

The following creates or edits the parameter file for an Extract group named `exte`:

```
EDIT PARAMS exte
```

4. Using the editing functions of the text editor, enter as many comment lines as you want to describe this file, making certain that each comment line is preceded with two hyphens (--).
5. On non-commented lines, enter the Oracle GoldenGate parameters, starting a new line for each parameter statement.

Oracle GoldenGate parameters have the following syntax:

```
PARAMETER_NAME argument [,option] [&]
```

Where:

- `PARAMETER_NAME` is the name of the parameter.
- `argument` is a required argument for the parameter. Some parameters take arguments, but others do not. Commas between arguments are optional.

```
EXTRACT exte
  USERIDALIAS ggadmin
  ENCRYPT AES192 KEYNAME mykey ENCRYPTTRAIL AES 192
  EXTTRAIL /north/ea, PURGE CUSEREXIT userexit.dll MyUserExit,
  INCLUDEUPDATEBEFORES, & PARAMS "init.properties"
  TABLE hr.employees;
```

- `[,option]` is an optional argument.

- `[&]` is required at the end of each line in a multi-line parameter statement, as in the `CUSEREXIT` parameter statement in the previous example. The exceptions are the following, which can accept, but do not require the ampersand because they terminate with a semicolon:

```

— MAP
— TABLE
— SEQUENCE
— FILE
— QUERY

```

6. Save and close the file.

Creating a Parameter File with a Text Editor

You can create a parameter file outside Admin Client by using a text editor, but make certain to:

- Save the parameter file with the name of the Extract or Replicat group that owns it. Use the `.prm` file extension. For example: `exte.prm`.
- Save the parameter file in the `dirprm` directory of the Oracle GoldenGate home directory.

Simplifying the Creation of Parameter Files

You can reduce the number of times that a parameter must be specified by using the following time-saving tools.

Using Wildcards

For parameters that accept object names, you can use asterisk (*) and question mark (?) wildcards. The use of wildcards reduces the work of specifying numerous object names or all objects within a given schema. For more information about using wildcards, see [Using Wildcards in Database Object Names](#).

Using OBEY

You can create a library of text files that contain frequently used parameter settings, and then you can call any of those files from the active parameter file by means of the `OBEY` parameter. The syntax for `OBEY` is:

```
OBEY file_name
```

Where:

file_name is the relative or full path name of the file.

Upon encountering an `OBEY` parameter in the active parameter file, Oracle GoldenGate processes the parameters from the referenced file and then returns to the active file to process any remaining parameters. `OBEY` is not supported for the `GLOBALS` parameter file.

If using the `CHARSET` parameter in a parameter file that includes an `OBEY` parameter, 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 (`\uX`) for all other multibyte specifications in the referenced file.

See *Parameters and Functions Reference for Oracle GoldenGate* for more information about OBEY.

See *Parameters and Functions Reference for Oracle GoldenGate* for more information about CHARSET.

Using Macros

You can use macros to automate multiple uses of a parameter statement. See [Simplify and Automate Work with Oracle GoldenGate Macros](#) .

Using Parameter Substitution

You can use parameter substitution to assign values to Oracle GoldenGate parameters automatically at run time, instead of assigning static values when you create the parameter file. That way, if values change from run to run, you can avoid having to edit the parameter file or maintain multiple files with different settings. You can simply export the required value at runtime. Parameter substitution can be used for any Oracle GoldenGate process.

To Use Parameter Substitution

1. For each parameter for which substitution is to occur, declare a runtime parameter instead of a value, and precede the runtime parameter name with a question mark (?) as shown in the following example.

```
SOURCEISFILE
EXTFILE ?EXTFILE
MAP hr. ?TABNAME, TARGET hr. TABNAME;
```

2. Before starting the Oracle GoldenGate process, use the shell of the operating system to pass the runtime values by means of an environment variable, as shown in the following examples:

Example 9-38 Parameter substitution on Windows

```
C:\> set EXTFILE=C:\ggs\extfile
C:\> set TABNAME=PROD.ACCOUNTS
C:\> replicat paramfile c:\ggs\dirprm\parmfl
```

Example 9-39 Parameter substitution on UNIX (korn shell)

```
$ EXTFILE=/ggs/extfile
$ export EXTFILE
$ TABNAME=PROD.ACCOUNTS
$ export TABNAME
$ replicat paramfile ggs/dirprm/parmfl
```

Validating a Parameter File

You can validate the parameter file from the Administration Service web interface. You can validate the Extract and Replicat parameters from the **Reports** tab. To access the **Reports** tab:

1. From Extract or Replicat section of the Administration Service Overview Page, click **Action** and then click **Details**.
2. Click the **Reports** tab to view the report for Extract and Replicat parameters, error log, and other information.

See [Access Extract Details](#) to learn how to check and edit the Extract parameters. See [Access Replicat Details](#) to learn about editing Replicat parameter files. Also see [Additional Parameters for Integrated Replicat](#)

You can also use the `checkprm` validation native command is run from the command line and give an assessment of the specified parameter file, with a configurable application and running environment. It can provide either a simple PASS/FAIL or with additional details about how the values of each parameter are stored and interpreted.

The `CHECKPRM` executable file can be found in the `$OGG_HOME/bin` directory of Microservices Architecture. See `checkprm` in the *Parameters and Functions Reference for Oracle GoldenGate*. The input to `checkprm` is case insensitive. If a value string contains spaces, it does not need to be quoted because `checkprm` can recognize meaningful values. If no mode is specified to `checkprm`, then all parameters applicable to any mode of the component will be accepted.

The output of `checkprm` is assembled with four possible sections:

- help messages
- pre-validation error
- validation result
- parameter details

A pre-validation error is typically an error that prevents a normal parameter validation from executing, such as missing options or an inaccessible parameter file. If an option value is specified incorrectly, a list of possible inputs for that option is provided. If the result is `FAIL`, each error is in the final result message. If the result is `PASS`, a message that some of the parameters are subject to further runtime validation. The parameter detailed output contains the validation context, and the specified parameters. The parameter and options are printed with proper indentation to illustrate these relationships.

See `CHECKPARAMS` parameter.

Simplify and Automate Work with Oracle GoldenGate Macros

You can use Oracle GoldenGate macros in parameter files to configure and reuse parameters, commands, and conversion functions. reducing the amount of text you must enter to do common tasks. A macro is a built-in automation tool that enables you to call a stored set of processing steps from within the Oracle GoldenGate parameter file. A macro can consist of a simple set of frequently used parameter statements to a complex series of parameter substitutions, calculations, or conversions. You can call other macros from a macro. You can store commonly used macros in a library, and then call the library rather than call the macros individually.

Oracle GoldenGate macros work with the following parameter files:

- `DEFGEN`
- `Extract`
- `Replicat`

There are two steps to using macros:

1. Defining a Macro
2. Calling a Macro

Define a Macro

To define an Oracle GoldenGate macro, use the `MACRO` parameter in the parameter file. `MACRO` defines any input parameters that are needed and it defines the work that the macro performs.

Syntax

```
MACRO #macro_name
PARAMS (#p1, #p2 [, ...])
BEGIN
macro_body
END;
```

Table 9-18 Macro Definition Arguments

Argument	Description
MACRO	Required. Indicates the start of an Oracle GoldenGate macro definition.
#macro_name	<p>The name of the macro. Macro and parameter names must begin with a macro character. The default macro character is the pound (#) character, as in <code>#macro1</code> and <code>#param1</code>.</p> <p>A macro or parameter name can be one word consisting of letters and numbers, or both. Special characters, such as the underscore character (<code>_</code>) or hyphen (<code>-</code>), can be used. Some examples of macro names are: <code>#mymacro</code>, <code>#macro1</code>, <code>#macro_1</code>, <code>#macro-1</code>, <code>#macro\$</code>. Some examples of parameter names are <code>#sourcecol</code>, <code>#s</code>, <code>#col1</code>, and <code>#col_1</code>.</p> <p>To avoid parsing errors, the macro character cannot be used as the first character of a macro name. For example, <code>##macro</code> is invalid. If needed, you can change the macro character by using the <code>MACROCHAR</code> parameter. See <i>Reference for Oracle GoldenGate for Windows and UNIX</i>.</p> <p>Macro and parameter names are not case-sensitive. Macro or parameter names within quotation marks are ignored.</p>
PARAMS (#p1, #p2)	Optional definition of input parameters. Specify a comma-separated list of parameter names and enclose it within parentheses. Each parameter must be referenced in the macro body where you want input values to be substituted. You can list each parameter on a separate line to improve readability (making certain to use the open and close parentheses to enclose the parameter list). See Call a Macro that Contains Parameters for more information.
BEGIN	Begins the macro body. Must be specified before the macro body.

Table 9-18 (Cont.) Macro Definition Arguments

Argument	Description
<i>macro_body</i>	<p>The macro body. The body is a syntax statement that defines the function that is to be performed by the macro. A macro body can include any of the following types of statements.</p> <ul style="list-style-type: none">• Simple parameter statements, as in: <code>COL1 = COL2</code>• Complex parameter statements with parameter substitution as in: <code>MAP #o.#t, TARGET #o.#t, KEYCOLS (#k), COLMAP (USEDEFAULTS);</code>• Invocations of other macros, as in: <code>#colmap (COL1, #sourcecol)</code>
END;	Ends the macro definition. The semicolon is required to complete the definition.

The following is an example of a macro definition that includes parameters. In this case, the macro simplifies the task of object and column mapping by supplying the base syntax of the MAP statement with input parameters that resolve to the names of the owners, the tables, and the KEYCOLS columns.

```
MACRO #macro1
PARAMS ( #o, #t, #k )
BEGIN
MAP #o.#t, TARGET #o.#t, KEYCOLS (#k), COLMAP (USEDEFAULTS);
END;
```

The following is an example of a macro that does not define parameters. It executes a frequently used set of parameters.

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

Call a Macro

To call a macro, use the following syntax where you want the macro to run within the parameter file.

Syntax

```
[target =] macro_name (val[, ...])

[target =] macro_name (val | {val, val, ...}[, ...])
```

Table 9-19 Syntax Elements for Calling a Macro

Argument	Description
<code>target =</code>	<p>Optional. Specifies the target to which the results of the macro are assigned or mapped. For example, <code>target</code> can be used to specify a target column in a COLMAP statement. In the following call to the <code>#make_date</code> macro, the column DATECOL1 is the target and will be mapped to the macro results.</p> <pre>DATECOL1 = #make_date (YR1, MO1, DAY1)</pre> <p>Without a target, the syntax to call <code>#make_date</code> is:</p> <pre>#make_date (YR1, MO1, DAY1)</pre>
<code>macro_name</code>	The name of the macro that is being called, for example: <code>#make_date</code> .
<code>(val[, ...])</code>	The parameter input values. This component is required whether or not the macro defines parameters. If the macro defines parameters, specify a comma-separated list of input values, in the order that corresponds to the parameter definitions in the <code>MACRO</code> parameter, and enclose the list within parentheses. If the macro does not define parameters, specify the open and close parentheses with nothing between them ().
<code>(val {val, val, ...})[, ...]</code>	The parameter input values. This component is required whether or not the macro defines parameters. If the macro defines parameters, specify a comma-separated list of input values, in the order that corresponds to the parameter definitions in the <code>MACRO</code> parameter, and enclose the list within parentheses. To pass multiple values to one parameter, separate them with commas and enclose the list within curly brackets. If the macro does not define parameters, specify the open and close parentheses with nothing between them ().

See the following topics to learn more about syntax for calling a macro:

Call a Macro that Contains Parameters

To call a macro that contains parameters, the call statement must supply the input values that are to be substituted for those parameters when the macro runs.

Valid input for a macro parameter is any of the following, preceded by the macro character (default is #):

- A single value in plain or quoted text, such as: `#macro (#name, #address, #phone)` or `#macro ("name", "address", "phone")`.

- A comma-separated list of values enclosed within curly brackets, such as: `#macro1 (SCOTT, DEPT, {DEPTNO1, DEPTNO2, DEPTNO3})`. The ability to substitute a block of values for any given parameter add flexibility to the macro definition and its usability in the Oracle GoldenGate configuration.
- Calls to other macros, such as: `#macro (#mycalc (col2, 100), #total)`. In this example, the `#mycalc` macro is called with the input values of `col2` and `100`.

Oracle GoldenGate substitutes parameter values within the macro body according to the following rules.

1. The macro processor reads through the macro body looking for instances of parameter names specified in the `PARAMS` statement.
2. For each occurrence of the parameter name, the corresponding parameter value specified during the call is substituted.
3. If a parameter name does not appear in the `PARAMS` statement, the macro processor evaluates whether or not the item is, instead, a call to another macro. (See [Calling Other Macros from a Macro](#).) If the call succeeds, the nested macro is executed. If it fails, the whole macro fails.

Example 9-40 Using Parameters to Populate a MAP Statement

The following macro definition specifies three parameter that must be resolved. The parameters substitute for the names of the table owner (parameter `#o`), the table (parameter `#t`), and the `KEYCOLS` columns (parameter `#k`) in a `MAP` statement.

```
MACRO #macro1 PARAMS ( #o, #t, #k ) BEGIN MAP #o.#t, TARGET #o.#t, KEYCOLS
(#k), COLMAP (USEDEFAULTS); END;
```

Assuming a table in the `MAP` statement requires only one `KEYCOLS` column, the following syntax can be used to call `#macro1`. In this syntax, the `#k` parameter can be resolved with only one value.

```
#macro1 (SCOTT, DEPT, DEPTNO1)
```

To call the macro for a table that requires two `KEYCOLS` columns, the curly brackets are used as follows to enclose both of the required values for the column names:

```
#macro1 (SCOTT, DEPT, {DEPTNO1, DEPTNO2})
```

The `DEPTNO1` and `DEPTNO2` values are passed as one argument to resolve the `#t` parameter. Tables with three or more `KEYCOLS` can also be handled in this manner, using additional values inside the curly brackets.

Example 9-41 Using a Macro to Perform Conversion

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

```
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;
```

The macro is called in the COLMAP clause:

```
MAP sales.acct_tab, TARGET sales.account,
COLMAP
(
  targcoll = sourcecoll,
  datecoll = #make_date(YR1, MO1, DAY1),
  datecol2 = #make_date(YR2, MO2, DAY2)
);
```

The macro expands as follows:

```
MAP sales.acct_tab, TARGET sales.account,
COLMAP
(
  targcoll = sourcecoll,
  datecoll = @DATE ('YYYY-MM-DD', 'CC', @IF (YR1 < 50, 20, 19), 'YY', YR1, 'MM',
MO1, 'DD', DAY1),
  datecol2 = @DATE ('YYYY-MM-DD', 'CC', @IF (YR2 < 50, 20, 19), 'YY', YR2, 'MM',
MO2, 'DD', DAY2)
);
```

Call a Macro without Input Parameters

To call a macro without input parameters, the call statement must supply the open and close parentheses, but without any input values: `#macro ()`.

The following macro is defined without input parameters. The body contains frequently used parameters.

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

This macro is called as follows:

```
#option_defaults ()
IGNOREUPDATES
MAP owner.srctab, TARGET owner.targtab;

#option_defaults ()
MAP owner.srctab2, TARGET owner.targtab2;
```

The macro expands as follows:

```
GETINSERTS
GETUPDATES
GETDELETES
INSERTDELETES
IGNOREUPDATES
MAP owner.srctab, TARGET owner.targetab;

GETINSERTS
GETUPDATES
GETDELETES
INSERTDELETES
MAP owner.srctab2, TARGET owner.targetab2;
```

Calling Other Macros from a Macro

To call other macros from a macro, create a macro definition similar to the following. In this example, the `#make_date` macro is nested within the `#assign_date` macro, and it is called when `#assign_date` runs.

The nested macro must define all, or a subset of, the same parameters that are defined in the base macro. In other words, the input values when the base macro is called must resolve to the parameters in both macros.

The following defines `#assign_date`:

```
MACRO #assign_date
PARAMS (#target_col, #year, #month, #day)
BEGIN
#target_col = #make_date (#year, #month, #day)
END;
```

The following defines `#make_date`. This macro creates a date format that includes a four-digit year, after first determining whether the two-digit input date should be prefixed with a century value of 19 or 20. Notice that the `PARAMS` statement of `#make_date` contains a subset of the parameters in the `#assign_date` macro.

```
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;
```

The following syntax calls `#assign_date`:

```
#assign_date (COL1, YEAR, MONTH, DAY)
```

The macro expands to the following given the preceding input values and the embedded `#make_date` macro:

```
COL1 = @DATE ('YYYY-MM-DD', 'CC', @IF (YEAR < 50, 20, 19), 'YY', YEAR, 'MM', MONTH, 'DD',
DAY)
```

Create Macro Libraries

You can create a macro library that contains one or more macros. By using a macro library, you can define a macro once and then use it within many parameter files.

To Create a Macro Library

1. Open a new file in a text editor.
2. Use commented lines to describe the library, if needed.
3. Use the following syntax to define each macro:

```
MACRO #macro_name
PARAMS (#p1, #p2 [, ...])
BEGIN
macro_body
END;
```

4. Save the file in the `dirprm` sub-directory of the Oracle GoldenGate directory as:

```
filename.mac
```

Where:

filename is the name of the file. The `.mac` extension defines the file as a macro library.

The following sample library named `datelib` contains two macros, `#make_date` and `#assign_date`.

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

MACRO #assign_date
PARAMS (#target_col, #year, #month, #day)
BEGIN
#target_col = #make_date (#year, #month, #day)
END;
```

To use a macro library, use the `INCLUDE` parameter at the beginning of a parameter file, as shown in the following sample Replicat parameter file.

```
INCLUDE /ggs/dirprm/datelib.mac
REPLICAT rep
ASSUMETARGETDEFS
USERIDALIAS ogg
MAP fin.acct_tab, TARGET fin.account;
```

When including a long macro library in a parameter file, you can use the `NOLIST` parameter to suppress the listing of each macro in the Extract or Replicat report file. Listing can be turned on and off by placing the `LIST` and `NOLIST` parameters anywhere within the parameter file or within the macro library file. In the following example, `NOLIST` suppresses the listing of each macro in the `hugelib` macro library. Specifying `LIST` after the `INCLUDE` statement restores normal listing to the report file.

```
NOLIST
INCLUDE /ggs/dirprm/hugelib.mac
LIST
INCLUDE /ggs/dirprm/mdatelib.mac
REPLICAT REP
```

Tracing Macro Expansion

You can trace macro expansion with the `CMDTRACE` parameter. With `CMDTRACE` enabled, macro expansion steps are shown in the Extract or Replicat report file.

Syntax

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

Where:

- `ON` enables tracing.
- `OFF` disables tracing.
- `DETAIL` produces a verbose display of macro expansion.

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

```
REPLICAT REP
MACRO #testmac
BEGIN
COL1 = COL2,
COL3 = COL4,
END;
...
CMDTRACE ON
MAP test.table1, TARGET test.table2,
COLMAP (#testmac);
CMDTRACE OFF
```

Using User Exits to Extend Oracle GoldenGate Capabilities

User exits are custom routines that you write in C programming code and call during Extract or Replicat processing. User exits extend and customize the functionality of the Extract and Replicat processes with minimal complexity and risk. With user exits, you can respond to database events when they occur, without altering production programs.

**Note:**

If you use `CUSEREXITS`, the `LD_LIBRARY_PATH` environment variable needs to be extended. By default, the `$OGG_HOME/lib` directory is part of the Oracle GoldenGate software Home directory. It is separated from the Deployment directory by design. If additional shared objects need to be added for User Exit functions, then it is recommended that you do not use the `$OGG_HOME/lib` directory and choose a different location. For `CUSEREXITS`, you must extend the `LD_LIBRARY_PATH` environment variable to a different location.

When to Implement User Exits

You can employ user exits as an alternative to, or in conjunction with, the column-conversion functions that are available within Oracle GoldenGate. User exits can be a better alternative to the built-in functions because a user exit processes data only once (when the data is extracted) rather than twice (once when the data is extracted and once to perform the transformation).

The following are some ways in which you can implement user exits:

- Perform arithmetic operations, date conversions, or table lookups while mapping from one table to another.
- Implement record archival functions offline.
- Respond to unusual database events in custom ways, for example by sending an e-mail message or a page based on an output value.
- Accumulate totals and gather statistics.
- Manipulate a record.
- Repair invalid data.
- Calculate the net difference in a record before and after an update.
- Accept or reject records for extraction or replication based on complex criteria.
- Normalize a database during conversion.

Making Oracle GoldenGate Record Information Available to the Routine

The basis for most user exit processing is the `EXIT_CALL_PROCESS_RECORD` function. For Extract, this function is called just before a record buffer is output to the trail. For Replicat, it is called just before a record is applied to the target. If source-target mapping is specified in the parameter file, the `EXIT_CALL_PROCESS_RECORD` event takes place after the mapping is performed.

When `EXIT_CALL_PROCESS_RECORD` is called, the record buffer and other record information are available to it through callback routines. The user exit can map, transform, clean, or perform any other operation with the data record. When it is finished, the user exit can return a status indicating whether the record should be processed or ignored by Extract or Replicat.

Creating User Exits

The following instructions help you to create user exits on Windows and UNIX systems. For more information about the parameters and functions that are described in these instructions, see Reference for Oracle GoldenGate for Windows and UNIX.



Note:

User exits are case-sensitive for database object names. Names are returned exactly as they are defined in the hosting database. Object names must be fully qualified.

To Create User Exits

1. In C code, create either a shared object (UNIX systems) or a DLL (Windows) and create or export a routine to be called from Extract or Replicat. This routine is the communication point between Oracle GoldenGate and your routines. Name the routine whatever you want. The routine must accept the following Oracle GoldenGate user exit parameters:
 - `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. This function enables you to use the `EXITPARAM` option of the `TABLE` or `MAP` statement to pass a parameter that is a literal string to the user exit. This is only valid during the exit call to process a specific record. This function also enables you to pass parameters specified with the `PARAMS` option of the `CUSEREXIT` parameter at the exit call startup.
2. In the source code, include the `usrdecs.h` file. The `usrdecs.h` file is the include file for the user exit API. It contains type definitions, return status values, callback function codes, and a number of other definitions. The `usrdecs.h` file is installed within the Oracle GoldenGate directory. Do not modify this file.
3. Include Oracle GoldenGate callback routines in the user exit when applicable. Callback routines retrieve record and application context information, and they modify the contents of data records. To implement a callback routine, use the `ERCALLBACK` function in the shared object. The user callback routine behaves differently based on the function code that is passed to the callback routine.

```
ERCALLBACK (function_code, buffer, result_code);
```

Where:

- `function_code` is the function to be executed by the callback routine.
- `buffer` is a void pointer to a buffer containing a predefined structure associated with the specified function code.
- `result_code` is the status of the function that is executed by the callback routine. The result code that is returned by the callback routine indicates whether or not the callback function was successful.
- On Windows systems, Extract and Replicat export the `ERCALLBACK` function that is to be called from the user exit routine. The user exit must explicitly load the callback function at run-time using the appropriate Windows API calls.

4. Include the `CUSEREXIT` parameter in your Extract or Replicat parameter file. This parameter accepts the name of the shared object or DLL and the name of the exported routine that is to be called from Extract or Replicat. You can specify the full path of the shared object or DLL or let the operating system's standard search strategy locate the shared object.

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

Where:

- `DLL` is a Windows DLL and `shared_object` is a UNIX shared object that contains the user exit function.
- `INCLUDEUPDATEBEFORES` gets before images for `UPDATE` operations.
- `PARAMS 'startup_string'` supplies a startup string, such as a startup parameter.

Example 9-42 Example of Base Syntax, UNIX

```
CUSEREXIT eruserexit.so MyUserExit
```

Example 9-43 Example Base Syntax, Windows

```
CUSEREXIT eruserexit.dll MyUserExit
```

Supporting Character-set Conversion in User Exits

To maintain data integrity, a user exit needs to understand the character set of the character-type data that it exchanges with an Oracle GoldenGate process. Oracle GoldenGate user exit logic provides globalization support for:

- character-based database metadata, such as the names of catalogs, schemas, tables, and columns
- the values of character-type columns, such as `CHAR`, `VARCHAR2`, `CLOB`, `NCHAR`, `NVARCHAR2`, and `NCLOB`, as well as string-based numbers, date-time, and intervals.

Properly converting between character sets allows column data to be compared, manipulated, converted, and mapped properly from one type of database and character set to another. Most of this processing is performed when the `EXIT_CALL_PROCESS_RECORD` call type is called and the record buffer and other record information is made available through callback routines.

The user exit has its own session character set. This is defined by the `GET_SESSION_CHARSET` and `SET_SESSION_CHARSET` callback functions. The caller process provides conversion between character sets if the character set of the user exit is different from the hosting context of the process.

To enable this support in user exits, there is the `GET_DATABASE_METADATA` callback function code. This function enables the user exit to get database metadata, such as the locale and the character set of the character-type data that it exchanges with the process that calls it (Extract, data pump, Replicat). It also returns how the database treats the case-sensitivity of object names, how it treats quoted and unquoted names, and how it stores object names.

For more information about these components, see Reference for Oracle GoldenGate for Windows and UNIX.

Using Macros to Check Name Metadata

The object name that is passed by the user exit API is the exact name that is encoded in the user-exit session character set, and exactly the same name that is retrieved from the database. If the user exit compares the object name with a literal string, the user exit must retrieve the database locale and then normalize the string so that it is compared with the object name in the same encoding.

Oracle GoldenGate provides the following macros that can be called by the user exit to check the metadata of database object names. For example, a macro can be used to check whether a quoted table name is case-sensitive and whether it is stored as mixed-case in the database server. These macros are defined in the `usrdecs.h` file.

Table 9-20 Macros for metadata checking

Macro	What it verifies
<code>supportsMixedCaseIdentifiers(nameMeta, DbObjType)</code>	Whether the database treats a mixed-case unquoted name of a specified data type as case-sensitive and stores the name in mixed case.
<code>supportsMixedCaseQuotedIdentifiers(nameMeta, DbObjType)</code>	Whether the database treats the mixed-case quoted name of a specified data type as case-sensitive and stores the name in mixed case.
<code>storesLowerCaseIdentifiers(nameMeta, DbObjType)</code>	Whether the database treats the mixed-case unquoted name of a specified data type as case-insensitive and stores the name in lower case.
<code>storesLowerCaseQuotedIdentifiers(nameMeta, DbObjType)</code>	Whether the database treats the mixed-case quoted name of a specified data type as case-insensitive and stores the name in lower case.
<code>storesMixedCaseIdentifiers(nameMeta, DbObjType)</code>	Whether the database treats the mixed-case unquoted name of a specified data type as case-insensitive and stores the name in mixed case.
<code>storesMixedCaseQuotedIdentifiers(nameMeta, DbObjType)</code>	Whether the database treats the mixed-case quoted name of a specified data type as case-insensitive and stores the name in mixed case.
<code>storesUpperCaseIdentifiers(nameMeta, DbObjType)</code>	Whether the database treats the mixed-case unquoted name of a specified data type as case-insensitive and stores the name in upper case.
<code>storesUpperCaseQuotedIdentifiers(nameMeta, DbObjType)</code>	Whether the database treats the mixed-case quoted name of a specified data type as case-insensitive and stores the name in upper case.

Describing the Character Format

The input parameter `column_value_mode` describes the character format of the data that is being processed and is used in several of the function codes. The following table describes the meaning of the `EXIT_FN_RAW_FORMAT`, `EXIT_FN_CHAR_FORMAT`, and `EXIT_FN_CNVTED_SESS_FORMAT` format codes, per data type.

Table 9-21 column_value_mode_matrix Meanings

Data Type	EXIT_FN_RAW_FORMAT	EXIT_FN_CHAR_FORMAT	EXIT_FN_CNVTED_SESS_FORMAT
CHAR "abc"	2-byte null indicator + 2-byte length info + column value 0000 0004 61 62 63 20	"abc" encoded in ASCII or EBCDIC. NULL terminated. Trailing spaces are trimmed.	"abc" encoded in user exit session character set. NOT NULL terminated. Trailing spaces are trimmed by default unless the GLOBALS parameter NOTRIMSPACES is specified.
NCHAR 0061 0062 0063 0020	2-byte null indicator + 2-byte length info + column value. 0000 0008 00 61 0062 0063 0020	"abc" (encoded in UTF8) or truncated at the first byte, depending on whether NCHAR is treated as UTF-8. NULL terminated. Trailing spaces are trimmed.	"abc" encoded in user exit session character set. NOT NULL terminated. Trailing spaces are trimmed by default unless the GLOBALS parameter NOTRIMSPACES is specified.
VARCHAR2 "abc"	2-byte null indicator + 2-byte length info + column value	"abc" encoded in ASCII or EBCDIC. NULL terminated. No trimming.	"abc" encoded in user exit session character set. NOT NULL terminated. No trimming.
NVARCHAR2 0061 0062 0063 0020	2-byte null indicator + 2-byte length info + column value	"abc" (encoded in UTF8) or truncated at the first byte, depending on whether NVARCHAR2 is treated as UTF-8. NULL terminated. No trimming.	"abc" encoded in user exit session character set. NOT NULL terminated. No trimming.
CLOB	2-byte null indicator + 2-byte length info + column value	Similar to VARCHAR2, but only output up to 4K bytes. NULL Terminated. No trimming.	Similar to VARCHAR2, but only output data requested in user exit session character set. NOT NULL terminated. No trimming.
NCLOB	2-byte null indicator + 2-byte length info + column value	Similar to NVARCHAR2, but only output up to 4K bytes. NULL terminated. No trimming.	Similar to NVARCHAR2, but only output data requested in user exit session character set. NOT NULL terminated. No trimming.
NUMBER 123.89	2-byte null indicator + 2-byte length info + column value	"123.89" encoded in ASCII or EBCDIC. NULL terminated.	"123.89" encoded in user exit session character set. NOT NULL terminated.
DATE 31-May-11	2-byte null indicator + 2-byte length info + column value	"2011-05-31" encoded in ASCII or EBCDIC. NULL terminated.	"2011-05-31" encoded in user exit session character set. NOT NULL terminated.
TIMESTAMP 31-May-11 12.00.00 AM	2-byte null indicator + 2-byte length info + column value	"2011-05-31 12.00.00 AM" encoded in ASCII or EBCDIC. NULL terminated.	"2011-05-31 12.00.00 AM" encoded in user exit session character set. NOT NULL terminated.

Table 9-21 (Cont.) column_value_mode_matrix Meanings

Data Type	EXIT_FN_RAW_FORMAT	EXIT_FN_CHAR_FORMAT	EXIT_FN_CNVTED_SESS_FORMAT
Interval Year to Month or Interval Day to Second	2-byte null indicator + 2-byte length info + column value	NA	NA
RAW	2-byte null indicator + 2-byte length info + column value	2-byte null indicator + 2-byte length info + column value	2-byte null indicator + 2-byte length info + column value

Upgrading User Exits

The `usrdecs.h` file is versioned to allow backward compatibility with existing user exits when enhancements or upgrades, such as new functions or structural changes, are added to a new Oracle GoldenGate release. The version of the `usrdecs.h` file is printed in the report file at the startup of Replicat or Extract.

To use new user exit functionality, you must recompile your routines to include the new `usrdecs` file. Routines that do not use new features do not need to be recompiled.

Viewing Examples of How to Use the User Exit Functions

Oracle GoldenGate installs the following sample user exit files into the `UserExitExamples` directory of the Oracle GoldenGate installation directory:

- `exitdemo.c` shows how to initialize the user exit, issue callbacks at given exit points, and modify data. It also demonstrates how to retrieve the fully qualified table name or a specific metadata part, such as the name of the catalog or container, or the schema, or just the unqualified table name. In addition, this demo shows how to process DDL data. The demo is not specific to any database type.
- `exitdemo_utf16.c` shows how to use UTF16-encoded data (both metadata and column data) in the callback structures for information exchanged between the user exit and the caller process.
- `exitdemo_more_recs.c` shows an example of how to use the same input record multiple times to generate several target records.
- `exitdemo_lob.c` shows an example of how to get read access to LOB data.
- `exitdemo_pk_befores.c` shows how to access the before and after image portions of a primary key update record, as well as the before images of regular updates (non-key updates). It also shows how to get target row values with `SQLEXEC` in the Replicat parameter file as a means for conflict detection. The resulting fetched values from the target are mapped as the target record when it enters the user exit.

Each directory contains the `*.c` files as well as makefiles and a `readme.txt` file.

10

Performance

Learn about different techniques available with Oracle GoldenGate to carry out performance monitoring and tuning activities.

Monitor

Learn about monitoring Oracle GoldenGate processes for performance and error handling.

Commands Used for Monitoring

You can view information about Extract and Replicat groups from the Oracle GoldenGate MA web interface at various levels. Another alternative is to use the command line interface to monitor various processes.

See [Monitor Processes from the Performance Metrics Service](#).

To learn about command syntax, usage, and examples, see the *Command Line Interface Reference* for Oracle GoldenGate.

Command	What it Shows
<code>INFO {EXTRACT REPLICAT} group [DETAIL]</code>	Run status, checkpoints, approximate lag, and environmental information. Displays the <code>INFO</code> output for all Oracle GoldenGate processes on the system.
<code>INFO ALL</code>	
<code>STATS {EXTRACT REPLICAT} group</code>	Displays statistics on processing volume, such as number of operations performed.
<code>STATUS {EXTRACT REPLICAT} group</code>	Displays the run status (starting, running, stopped, abended) for Extract and Replicat processes.
<code>LAG {EXTRACT REPLICAT} group</code>	Displays the latency between last record processed and timestamp in the data source.
<code>INFO {EXTTRAIL RMTTRAIL } trail</code>	Displays the name of associated process, position of last data processed, maximum file size.
<code>SEND {EXTRACT REPLICAT } group</code>	Depending on the process and selected options, returns information about memory pool, lag, TCP statistics, long-running transactions, process status, recovery progress, and more.

Command	What it Shows
VIEW REPORT <i>group</i>	Shows contents of the discard file or process report.
VIEW GGSEVT	Shows contents of the Oracle GoldenGate error log.
COMMAND ER <i>wildcard</i>	<p>Information dependent on the COMMAND type:</p> <p>INFO</p> <p>LAG</p> <p>SEND</p> <p>STATS</p> <p>STATUS</p> <p><i>wildcard</i></p> <p>is a wildcard specification for the process groups to be affected, for example:</p> <p>INFO ER <i>ext*</i></p> <p>STATS ER *</p>
INFO PARAM	Queries for and displays static information.
GETPARAMINFO	Displays currently-running parameter values.
INFO DISTPATH	Returns information about distribution paths. Before you run this command, ensure that the Distribution Service is running for that deployment.
INFO EXTTRAIL	Retrieves 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.

Command	What it Shows
INFO RMTTRAIL	Retrieves 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.
INFO ER	Retrieves information on multiple Extract and Replicat groups as a unit.
INFO CHECKPOINTTABLE	Confirms the existence of a checkpoint table and view the date and time that it was created.
INFO CREDENTIALS	Retrieves a list of credentials.
INFO ENCRYPTIONPROFILE	Returns information about the encryption profiles available with the Service Manager.
INFO HEARTBEATTABLE	Displays information about the heartbeat tables configured in the database.
INFO AUTHORIZATIONPROFILE	Lists all the authorization profiles in a deployment or information on a specific authorization profile for a specific deployment.
INFO MASTERKEY	Displays the contents of a currently open master-key wallet. If a wallet store does not exist, a new wallet store file is created. This wallet store file is then used to host different encrypted keys as they are created.
INFO PROFILE	Returns information about managed process profiles.
INFO RECVPATH	Returns information about a target-initiated distribution path in the Receiver Service. Before you run this command, ensure that the Receiver Service is running.
INFO SCHEMATRANDATA	Valid for Oracle database only. Determine whether Oracle schema-level supplemental logging is enabled for the specified schema or if any instantiation information is available. Use the <code>DBLOGIN</code> command to establish a database connection before using this command.
INFO TRACETABLE	Verifies the existence of the specified trace table in the local instance of the database.
INFO TRANDATA	Displays different outputs depending on the database.

Command	What it Shows
<code>STATS DISTPATH RECVPATH</code>	Get the statistics for the distribution path (DISTPATH) or receiver path (RECVPATH).
<code>STATS ER</code>	Retrieve 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.
<code>STATUS ER</code>	Checks the status of multiple Extract and Replicat groups as a unit.
<code>STATUS DEPLOYMENT</code>	View the status of the specified deployment.
<code>STATUS PMSRVR</code>	Status of Performance Service.
<code>STATUS SERVICE</code>	Displays the status of specified Oracle GoldenGate service.

Monitor Processes from the Performance Metrics Service

The Performance Metrics Service uses the metrics service to collect and store instance deployment performance results. When you arrive at the Performance Metrics Service Overview page, you see all the Oracle GoldenGate processes in their current state. You can click a process to view its performance metrics. You can also access service messages and status change details from this page.

Here's a general overview of the tasks that you can perform from this page.

Task	Description
Review Messages	Review Messages from the Messages Overview tab.
Review Status Changes	Click the Review Status Changes tab to review changes in status of a service.

Review Messages from Messages Tab

Messages from the Services are displayed in Performance Metrics Service Overview page.

To review the messages sent or received, do the following:

1. From the Service Manager, click **Performance Metrics Service**.
2. On the Performance Metrics Service home page, click the **Messages Overview** tab (if it's not already selected) to see a drill down into all the service messages.

Scroll through the list of messages or search for a specific message by entering the text in the message.

- Click **Refresh** to get a synchronized real-time list of messages before you start searching. You can also change the page size to view more or fewer messages.

Review Status Changes

Real-time status changes to microservices can be monitored from the Performance Metrics Service Status Changes Overview tab.

Status change messages show the date, process name, and its status, which could be running, starting, stopped, or killed.

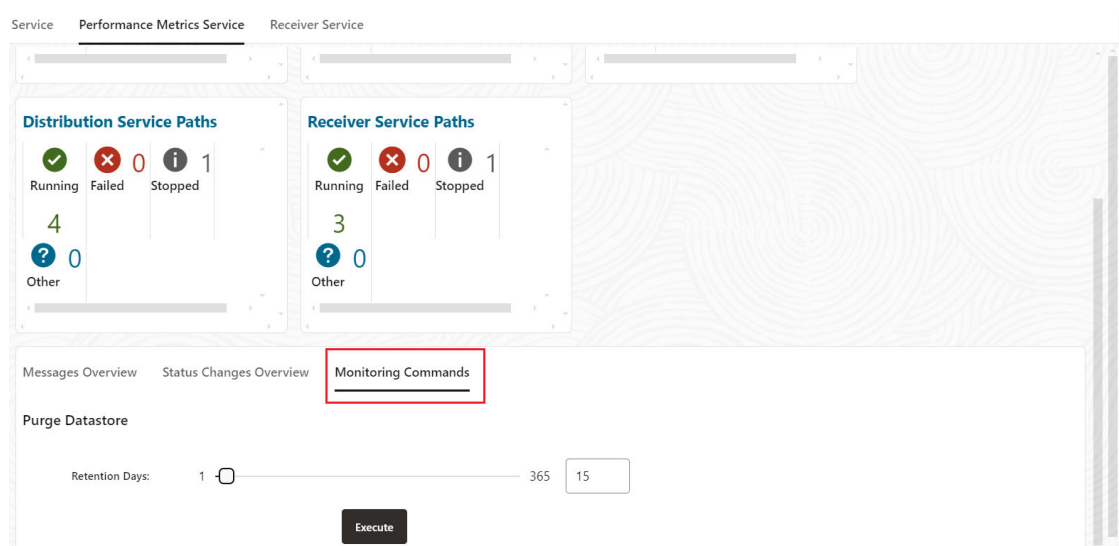
To view status changes, click **Performance Metrics Service** from the Service Manager home page, and then click the **Status Changes Overview** tab. A list of status change messages from the service appears.

If you are searching for specific messages, you can use the search but make sure you click **Refresh** before you search to ensure that you get the updated status for services.

Note that the search messages appear in different colors to differentiate critical and informational messages.

Purge Datastore

You can change the datastore retention and purge it from the Performance Metrics Service **Monitoring Commands** tab, as shown in the following image:



To view status changes, click Performance Metrics Service from the Service Manager home page, and then click the **Monitoring Commands** tab.

The current process retention (in days) is displayed.

You can enter the number of retention days or use the sliding icon to set the new period from 1 to 365 days, then **Execute** to activate the purge. The details of the purge are also displayed.

Protocols for Performance Monitoring for Different Operating Systems

Oracle GoldenGate uses Unix Domain Sockets (UDS) for UNIX-based and Named Pipes (for Windows) techniques to send monitoring points from Extract, Replicat, and other processes to the Performance Monitoring Service of the deployment.

For each deployment, the Performance Metrics Service is local to the host. This makes it more secure to use the Unix Domain Sockets (UDS) protocol or Named Pipes technique in Windows for Inter-process Communication (IPC) with the service and improve overall performance. Named Pipes utilizes a unique file system called NPFS (Named Pipe filesystem) that allows managing the security as any file subject using security checks for file access.

- UDS is available with Oracle and non-Oracle databases. The UDS file is located in the `$OGG_HOME/var/temp` directory of the deployment.
- The standard location for named pipes in Windows is `\\ServerName\pipe\PipeName` (`\\ServerName\pipe\`).

Monitor an Extract Recovery

If Extract abends when a long-running transaction is open, it can seem to take a long time to recover when it is started again. To recover its processing state, Extract must search back through the online and archived logs (if necessary) to find the first log record for that long-running transaction. The farther back in time that the transaction started, the longer the recovery takes, in general, and Extract can appear to be stalled.

To confirm that Extract is recovering properly, use the `SEND EXTRACT` command with the `STATUS` option. One of the following status notations appears, and you can follow the progress as Extract changes its log read position over the course of the recovery.

In recovery[1]

Extract is recovering to its checkpoint in the transaction log. This implies that it is reading from either the BR checkpoint files and then archived/online logs, or reading from Recovery Checkpoint in archived/online log.

In recovery[2]

Extract is recovering from its checkpoint to the end of the trail. This implies that a recovery marker is appended to the output trail when the last transaction was not completely written then rewriting the transaction.

Recovery complete

The recovery is finished, and normal processing will resume.

Monitor Lag

Lag statistics show you how well the Oracle GoldenGate processes are keeping pace with the amount of data that is being generated by the business applications. With this information, you can diagnose suspected problems and tune the performance of the Oracle GoldenGate processes to minimize the latency between the source and target databases.

About Lag

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.

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.

To view lag statistics, use either the `LAG` or `SEND ER`, `SEND EXTRACT`, `SEND REPLICAT` commands.

**Note:**

The `INFO` command also returns a lag statistic, but this statistic is taken from the last record that was checkpointed, not the current record that is being processed. It is less accurate than `LAG` or `INFO`.

Monitor Lag Using Automatic Heartbeat Tables

You can use the default automatic heartbeat table functionality to monitor end-to-end replication lag. Automatic heartbeats are sent from each source database into the replication streams, by updating the records in a *heartbeat seed table* and a *heartbeat table*, and constructing a *heartbeat history table*. Each of the replication processes in the replication path process these heartbeat records and update the information in them. These heartbeat records are inserted or updated into the heartbeat table at the target databases.

The heartbeat tables contain the following information:

- Source database
- Destination database
- Information about the outgoing replication streams:
 - Names of the Extract, Distribution Service, and or Replicat processes in the path
 - Timestamps when heartbeat records were processed by the replication processes.
- Information about the incoming replication streams:
 - Names of the Extract, Distribution Service, and or Replicat processes in the path
 - Timestamps when heartbeat records were processed by the replication processes.

Using the information in the heartbeat table and the heartbeat history table, the current and historical lags in each of the replication can be computed.

Replicat can track the current restart position of Extract with automatic heartbeat tables (`LOGBSN`). This allows regenerating the trail files from the source database, if required and minimizes the redo log retention period of the source database. Also, by tracking the most recent Extract restart position, the tombstone tables for automatic Conflict Detection and Resolution (`ACDR`) tables can be purged more frequently.

In a bidirectional configuration, the heartbeat table has as many entries as the number of replication paths to neighbors that the database has and in a unidirectional setup, the table at the source is empty. The outgoing columns have the timestamps and the outgoing path, the local Extract and the downstream processes. The incoming columns have the timestamps and path of the upstream processes and local Replicat.

In a unidirectional configuration, the target database will populate only the incoming columns in the heartbeat table.

**Note:**

The Automatic Heartbeat functionality is not supported on MySQL version 5.5.

Monitor an Extract Recovery

If Extract abends when a long-running transaction is open, it can seem to take a long time to recover when it is started again. To recover its processing state, Extract must search back through the online and archived logs (if necessary) to find the first log record for that long-running transaction. The farther back in time that the transaction started, the longer the recovery takes, in general, and Extract can appear to be stalled.

To confirm that Extract is recovering properly, use the `SEND EXTRACT` command with the `STATUS` option. One of the following status notations appears, and you can follow the progress as Extract changes its log read position over the course of the recovery.

In recovery[1]

Extract is recovering to its checkpoint in the transaction log. This implies that it is reading from either the BR checkpoint files and then archived/online logs, or reading from Recovery Checkpoint in archived/online log.

In recovery[2]

Extract is recovering from its checkpoint to the end of the trail. This implies that a recovery marker is appended to the output trail when the last transaction was not completely written then rewriting the transaction.

Recovery complete

The recovery is finished, and normal processing will resume.

Heartbeat Table End-To-End Replication Flow

The end-to-end replication process for heartbeat tables relies on using the Oracle GoldenGate trail format. The process is as follows:

Add a heartbeat table to each of your databases with the `ADD HEARTBEATTABLE` command. Add the heartbeat table to all source and target instances and then restart existing Oracle GoldenGate processes to enable heartbeat functionality. Depending on the database, you may or may not be required to create or enable a job to populate the heartbeat table data. See the following sample:

```
DBLOGIN USERIDALIAS alias [DOMAIN domain][SYSDBA | SQLID sqlid]
[SESSIONCHARSET character_set]
```

```
ADD HEARTBEATTABLE
```

(Optional) For Oracle Databases, you must ensure that the Oracle `DBMS_SCHEDULER` is operating correctly as the heartbeat update relies on it. You can query the `DBMS_SCHEDULER` by issuing:

```
SELECT START_DATE, LAST_START_DATE, NEXT_RUN_DATE
FROM DBA_SCHEDULER_JOBS
```

Where `job_name = 'GG_UPDATE_HEARTBEATS'`;

Then look for valid entries for `NEXT_RUN_DATE`, which is the next time the scheduler will run. If this is a timestamp in the past, then no job will run and you must correct it.

A common reason for the scheduler not working is when the parameter `job_queue_processes` is set too low (typically zero). Increase the number of `job_queue_processes` configured in the database with the `ALTER SYSTEM SET JOB_QUEUE_PROCESSES = ##;` command where `##` is the number of job queue processes.

Run an Extract, which on receiving the logical change records (LCR) checks the value in the `OUTGOING_EXTRACT` column.

- If the Extract name matches this value, the `OUTGOING_EXTRACT_TS` column is updated and the record is entered in the trail.
- If the Extract name does not match then the LCR is discarded.
- If the `OUTGOING_EXTRACT` value is `NULL`, it is populated along with `OUTGOING_EXTRACT_TS` and the record is entered in the trail.

The Distribution Service on reading the record, checks the value in the `OUTGOING_ROUTING_PATH` column. This column has a list of distribution paths.

If the value is `NULL`, then the column is updated with the current group name (and path if this is a Distribution Service), `"*"`, update the `OUTGOING_ROUTING_TS` column, and the record is written into its target trail file.

If the value has a `"*"` in the list, then replace it with `group_name[:pathname], "*"'`, update the `OUTGOING_ROUTING_TS` column, and the record is written into its target trail file. When the value does not have an asterisk (*) in the list and the distribution path name is in the list, then the record is sent to the path specified in the relevant `group_name[:pathname], "*"'` pair in the list. If the distribution path name is not in the list, then the record is discarded.

Run a Replicat, which on receiving the record checks the value in the `OUTGOING_REPLICAT` column.

- If the Replicat name matches the value, the row in the heartbeat table is updated and the record is inserted into the history table.
- If the Replicat name does not match, the record is discarded.
- If the value is `NULL`, the row in the heartbeat and heartbeat history tables are updated with an implicit invocation of the Replicat column mapping.

Automatic Replicat Column Mapping:

<code>REMOTE_DATABASE</code>	<code>= LOCAL_DATABASE</code>
<code>INCOMING_EXTRACT</code>	<code>= OUTGOING_EXTRACT</code>
<code>INCOMING_ROUTING_PATH</code>	<code>= OUTGOING_ROUTING_PATH with "*" removed</code>
<code>INCOMING_REPLICAT</code>	<code>= @GETENV ("GGENVIRONMENT", "GROUPNAME")</code>

```

INCOMING_HEARTBEAT_TS      = HEARTBEAT_TIMESTAMP
INCOMING_EXTRACT_TS       = OUTGOING_EXTRACT_TS
INCOMING_ROUTING_TS       = OUTGOING_ROUTING_TS
INCOMING_REPLICAT_TS      = @DATE ('UYYYY-MM-DD
HH:MI:SS.FFFFFFFF','JTSLCT',@GETENV ('JULIANTIMESTAMP'))
LOCAL_DATABASE            = REMOTE_DATABASE
OUTGOING_EXTRACT          = INCOMING_EXTRACT
OUTGOING_ROUTING_PATH     = INCOMING_ROUTING_PATH
OUTGOING_HEARTBEAT_TS     = INCOMING_HEARTBEAT_TS
OUTGOING_REPLICAT         = INCOMING_REPLICAT
OUTGOING_HEARTBEAT_TS     = INCOMING_HEARTBEAT_TS

```

Additional Considerations:

Computing lags as the heartbeat flows through the system relies on the clocks of the source and target systems to be set up correctly. It is possible that the lag can be negative if the target system is ahead of the source system. The lag is shown as a negative number so that you are aware of their clock discrepancy and can take actions to fix it.

The timestamp that flows through the system is in UTC. There is no time zone associated with the timestamp so when viewing the heartbeat tables, the lag can be viewed quickly even if different components are in different time zones. You can write any view you want on top of the underlying tables; UTC is recommended.

All the heartbeat entries are written to the trail in UTF-8.

The outgoing and incoming paths together uniquely determine a row. Meaning that if you have two rows with same outgoing path and a different incoming path, then it is considered two unique entries.

Heartbeat Table Details

The `GG_HEARTBEAT` table displays timestamp information of the end-to-end replication time and the timing information at the different components primary and secondary Extract and Replicat.

In a unidirectional environment, only the target database contains information about the replication lag. That is the time when a record is generated at the source database and becomes visible to clients at the target database.



Note:

The automatic heartbeat tables don't populate the `OUTGOING_*` columns with data, when both the source and remote databases have the same name. To change the database name, use the utility `DBNEWID`. For details, see the [DBNEWID Utility](#).

Column	Data Type	Description
LOCAL_DATABASE	VARCHAR2	Local database where the replication time from the remote database is measured.
HEARTBEAT_TIMESTAMP	TIMESTAMP (6)	The point in time when a timestamp is generated at the remote database.
REMOTE_DATABASE	VARCHAR2	Remote database where the timestamp is generated

Column	Data Type	Description
INCOMING_EXTRACT	VARCHAR2	Name of the primary Extract (capture) at the remote database
INCOMING_ROUTING_PATH	VARCHAR2	Name of the secondary Extract (pump) at the remote database
INCOMING_REPLICAT	VARCHAR2	Name of the Replicat on the local database.
INCOMING_HEARTBEAT_TS	TIMESTAMP (6)	Final timestamp when the information is inserted into the GG_HEARTBEAT table at the local database.
INCOMING_EXTRACT_TS	TIMESTAMP (6)	Timestamp of the generated timestamp is processed by the primary Extract at the remote database.
INCOMING_ROUTING_TS	TIMESTAMP (6)	Timestamp of the generated timestamp is processed by the secondary Extract at the remote database.
INCOMING_REPLICAT_TS	TIMESTAMP (6)	Timestamp of the generated timestamp is processed by Replicat at the local database.
OUTGOING_EXTRACT	VARCHAR2	Bidirectional/N-way replication: Name of the primary Extract on the local database.
OUTGOING_ROUTING_PATH	VARCHAR2	Bidirectional/N-way replication: Name of the secondary Extract on the local database.
OUTGOING_REPLICAT	VARCHAR2	Bidirectional/N-way replication: Name of the Replicat on the remote database.
OUTGOING_HEARTBEAT_TS	TIMESTAMP (6)	Bidirectional/N-way replication: Final timestamp when the information is inserted into the table at the remote database.
OUTGOING_EXTRACT_TS	TIMESTAMP (6)	Bidirectional/N-way replication: Timestamp of the generated timestamp is processed by the primary Extract on the local database.
OUTGOING_ROUTING_TS	TIMESTAMP (6)	Bidirectional/N-way replication: Timestamp of the generated timestamp is processed by the secondary Extract on the local database.
OUTGOING_REPLICAT_TS	TIMESTAMP (6)	Bidirectional/N-way replication: Timestamp of the generated timestamp is processed by Replicat on the remote database.
INCOMING_REPLICAT_LW_CSN	VARCHAR2	-
INCOMING_EXTRACT_HEARTBEAT_CSN	VARCHAR2	-

Column	Data Type	Description
INCOMING_EXTRACT_RESTART_C SN	VARCHAR2	-
INCOMING_EXTRACT_RESTART_T S	TIMESTAMP (6)	-

The `GG_HEARTBEAT_HISTORY` table displays historical timestamp information of the end-to-end replication time and the timing information at the different components primary and secondary Extract and Replicat.

In a unidirectional environment, only the destination database contains information about the replication lag.

Timestamps are managed in UTC time zone. That is the time when a record is generated at the source database and becomes visible to clients at the target database.

Column	Data Type	Description
LOCAL_DATABASE	VARCHAR2	Local database where the end-to-end lag is measured.
HEARTBEAT_RECEIVED_TS	TIMESTAMP (6)	Point in time when a timestamp from the remote database receives at the local database.
REMOTE_DATABASE	VARCHAR2	Remote database where the timestamp is generated.
INCOMING_EXTRACT	VARCHAR2	Name of the primary Extract on the remote database.
INCOMING_ROUTING_PATH	VARCHAR2	Name of the secondary Extract of the remote database.
INCOMING_REPLICAT	VARCHAR2	Name of the Replicat on the local database.
INCOMING_HEARTBEAT_TS	TIMESTAMP (6)	Final timestamp when the information is inserted into the <code>GG_HEARTBEAT_HISTORY</code> table on the local database.
INCOMING_EXTRACT_TS	TIMESTAMP (6)	Timestamp when the generated timestamp is processed by the primary Extract on the remote database.
INCOMING_ROUTING_TS	TIMESTAMP (6)	Timestamp when the generated timestamp is processed by the secondary Extract on the remote database.
INCOMING_REPLICAT_TS	TIMESTAMP (6)	Timestamp when the generated timestamp is processed by Replicat on the local database.
OUTGOING_EXTRACT	VARCHAR2	Bidirectional/N-way replication: Name of the primary Extract from the local database.
OUTGOING_ROUTING_PATH	VARCHAR2	Bidirectional/N-way replication: Name of the secondary Extract from the local database.

Column	Data Type	Description
OUTGOING_REPLICAT	VARCHAR2	Bidirectional/N-way replication: Name of the Replicat on the remote database.
OUTGOING_HEARTBEAT_TS	TIMESTAMP (6)	Bidirectional/N-way replication: Final timestamp when the information is persistently inserted into the table of the remote database.
OUTGOING_EXTRACT_TS	TIMESTAMP (6)	Bidirectional/N-way replication: Timestamp when the generated timestamp is processed by the primary Extract on the local database.
OUTGOING_ROUTING_TS	TIMESTAMP (6)	Bidirectional/N-way replication: Timestamp when the generated timestamp is processed by the secondary Extract on the local database.
OUTGOING_REPLICAT_TS	TIMESTAMP (6)	Bidirectional/N-way replication: Timestamp when the generated timestamp is processed by Replicat on the remote database.
REPLICAT_LOW_WATERMARK_CSN	String	This column is populated by Replicat when it processes this heartbeat record. It populates this column with its current low watermark (LWM) when it processes this record. This allows us to choose a LOGBSN from a heartbeat record which is as of the Replicat LWM.
SOURCE_EXTRACT_HEARTBEAT_CSN	String	This column is populated by Extract and contains the source commit SCN for the heartbeat transaction in the source database. The heartbeat job on the source database cannot populate this value as it will not know the commit SCN apriori.
SOURCE_EXTRACT_RESTART_CSN	String	This column will be populated by Extract and will contain the current LOGBSN when Extract processes this particular heartbeat record. The heartbeat job on the source database will not populate this value.
SOURCE_EXTRACT_RESTART_CSN_TS	TIMESTAMP	This column will be populated by Extract and will contain the redo timestamp in UTC that corresponds to the current LOGBSN when Extract processes this particular heartbeat record. The heartbeat job on the source database will not populate this value.

The `GG_LAG` view displays information about the replication lag between the local and remote databases.

In a unidirectional environment, only the destination database contains information about the replication lag. The lag is measured in seconds.

Column	Data Type	Description
LOCAL_DATABASE	VARCHAR2	Local database where the end-to-end replication lag from the remote database is measured.
CURRENT_LOCAL_TS	TIMESTAMP (6)	Current timestamp of the local database.
REMOTE_DATABASE	VARCHAR2	Remote database where the timestamp is generated.
INCOMING_HEARTBEAT_AGE	NUMBER	The age of the most recent heartbeat received from the remote database.
INCOMING_PATH	VARCHAR2	Replication path from the remote database to the local database with Extract and Replicat components.
INCOMING_LAG	NUMBER	Replication lag from the remote database to the local database. This is the time where the heartbeat where generated at the remote database minus the time where the information was persistently inserted into the table at the local database.
OUTGOING_HEARTBEAT_AGE	NUMBER	The age of the most recent heartbeat from the local database to the remote database.
OUTGOING_PATH	VARCHAR2	Replication Path from Local database to the remote database with Extract and Replicat components
OUTGOING_LAG	NUMBER	Replication Lag from the local database to the remote database. This is the time where the heartbeat where generated at the local database minus the time where the information was persistently inserted into the table at the remote database.
REMOTE_EXTRACT_RESTART_CSN	String	Source Extract restart position.
REMOTE_DATABASE DB_UNIQUE_NAME	String	Remote database unique name is displayed. If no unique name exists, then the <code>DB_NAME</code> value is displayed.
REMOTE_EXTRACT_RESTART_CSN _TIME	Timestamp	Timestamp associated with source Extract redo position.

Column	Data Type	Description
REMOTE_DB_OLDEST_OPEN_TXN_AGE	Timestamp	Age of the oldest open transaction at the source database that Extract is currently processing. This column can be calculated as <code>SYSTIMESTAMP - REMOTE_EXTRACT_RESTART_TIME</code> .
LOCAL_REPLICAT_LWM_CSN	String	Low watermark CSN of the local Replicat when it processed the heartbeat.

The `GG_LAG_HISTORY` view displays the history information about the replication lag history between the local and remote databases.

In a unidirectional environment, only the destination database contains information about the replication lag.

The unit of the lag units is in seconds.

Column	Data Type	Description
LOCAL_DATABASE	VARCHAR2	Local database where the end-to-end replication lag from the remote database is measured.
HEARTBEAT_RECEIVED_TS	TIMESTAMP (6)	Point in time when a timestamp from the remote database receives on the local database.
REMOTE_DATABASE	VARCHAR2	Remote database where the timestamp is generated.
DB_NAME	String	Remote database name.
DB_UNIQUE_NAME	String	Remote database unique name. If the database unique name doesn't exist, then the <code>DB_NAME</code> and <code>DB_UNIQUE_NAME</code> will be same. In a switchover to standby scenario, the <code>db_unique_name</code> will change but the <code>db_name</code> and replication path remain the same.
INCOMING_HEARTBEAT_AGE	NUMBER	The age of the heartbeat table.
INCOMING_PATH	VARCHAR2	Replication path from the remote database to local database with Extract and Replicat components.
INCOMING_LAG	NUMBER	Replication lag from the remote database to the local database. This is the time where the heartbeat was generated at the remote database minus the time where the information was persistently inserted into the table on the local database.
OUTGOING_HEARTBEAT_AGE	NUMBER	

Column	Data Type	Description
OUTGOING_PATH	VARCHAR2	Replication path from local database to the remote database with Extract and Replicat components.
OUTGOING_LAG	NUMBER	Replication lag from the local database to the remote database. This is the time where the heartbeat was generated at the local database minus the time where the information was persistently inserted into the table on the remote database.
REMOTE_EXTRACT_RESTART_CSN	String	Source Extract restart position.
REMOTE_EXTRACT_RESTART_CSN_TIMESTAMP	TIMESTAMP	Timestamp associated with source Extract redo position.
REMOTE_DB_OLDEST_OPEN_TXN_AGE	TIMESTAMP	Age of the oldest open transaction at the source database that Extract is currently processing. This column can be calculated as: <code>SYSTIMESTAMP - REMOTE_EXTRACT_RESTART_TIMESTAMP</code>
LOCAL_REPLICAT_LWM_CSN	String	Low watermark CSN of the local Replicat when it processed the heartbeat.
INCOMING_EXTRACT_LAG	-	-
INCOMING_ROUTINE_LAG	-	-
INCOMING_REPLICAT_READ_LAG	-	-
INCOMING_REPLICAT_LAG	-	-
OUTGOING_EXTRACT_LAG	-	-
OUTGOING_ROUTINE_LAG	-	-
OUTGOING_REPLICAT_READ_LAG	-	-
OUTGOING_REPLICAT_LAG	-	-

Update Heartbeat Tables

The `HEARTBEAT_TIMESTAMP` column in the heartbeat seed table must be updated periodically by a database job. The default heartbeat interval is 1 minute and this interval can be specified or overridden using from the command line or the Administration Service web interface.

For Oracle Database, the database job is created automatically.

For all other supported databases, you must create background jobs to update the heartbeat timestamp using the database specific scheduler functionality.

See `ADD HEARTBEATTABLE`, `ALTER HEARTBEATTABLE` for details on updating the heartbeat table.

Purge the Heartbeat History Tables

The heartbeat history table is purged periodically using a job. The default interval is 30 days and this interval can be specified or overridden using a command line interface such as Admin Client or the Administration Service web interface.

For Oracle Database, the database job is created automatically.

For all other supported databases, you must create background jobs to purge the heartbeat history table using the database specific scheduler functionality.

Best Practice

Oracle recommends that you:

- Use the same heartbeat frequency on all the databases to makes diagnosis easier.
- Adjust the retention period if space is an issue.
- Retain the default heartbeat table frequency; the frequency set to be 30 to 60 seconds gives the best results for most workloads.
- Use lag history statistics to collect lag and age information.

Using the Automatic Heartbeat Commands

You can use the heartbeat table commands to control the Oracle GoldenGate automatic heartbeat functionality as follows.

Command	Description
ADD HEARTBEATTABLE	Creates the heartbeat tables required for automatic heartbeat functionality including the LOGBSN columns.
ALTER HEARTBEATTABLE	Alters existing heartbeat objects.
ALTER HEARTBEATTABLE UPGRADE	Alters the heartbeat tables to add the LOGBSN columns to the heartbeat tables. This is optional.
DELETE HEARTBEATTABLE	Deletes existing heartbeat objects.
DELETE HEARTBEATENTRY	Deletes entries in the heartbeat table.
INFO HEARTBEATTABLE	Displays heartbeat table information.

Db2 z/OS: Interpret Statistics for Update Operations

The actual number of DML operations that are executed on the Db2 database might not match the number of extracted DML operations that are 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.

Monitor Processing Volume

The `STATS` commands show you the amount of data that is being processed by an Oracle GoldenGate process, and how fast it is being moved through the Oracle GoldenGate system. With this information, you can diagnose suspected problems and tune the performance of the Oracle GoldenGate processes. These commands provide a variety of options to select and filter the output.

The `STATS` commands are: `STATS EXTRACT`, `STATS REPLICAT`, or `STATS ER` command.

You can send interim statistics to the report file at any time with the `SEND EXTRACT` or `SEND REPLICAT` command with the `REPORT` option.

Use the Error Log

Use the Oracle GoldenGate error log to view:

- a history of commands
- Oracle GoldenGate processes that started and stopped
- processing that was performed
- errors that occurred
- informational and warning messages

Because the error log shows events as they occurred in sequence, it is a good tool for detecting the cause (or causes) of an error. For example, you might discover that:

- someone stopped a process
- a process failed to make a TCP/IP or database connection
- a process could not open a file

To view the error log, use any of the following:

- Standard shell command to view the `ggserr.log` file within the root Oracle GoldenGate directory
- `VIEW GGSEVT` command.

You can control the `ggserr.log` file behavior to:

- Roll over the file when it reaches a maximum size, which is the default to avoid disk space issues.
- All messages are appended to the file by all processes without regard to disk space.
- Disable the file.
- Route messages to another destination, such as the system log.

This behavior is controlled and described in the `ogg-ggserr.xml` file in one of the following locations:

Microservices Architecture
`$OGG_HOME/etc/conf/logging/`

Use the Process Report

Use the process report to view (depending on the process):

- parameters in use
- table and column mapping
- database information
- runtime messages and errors
- runtime statistics for the number of operations processed

Every Extract, Replicat process generates a report file. The report can help you diagnose problems that occurred during the run, such as invalid mapping syntax, SQL errors, and connection errors.

To view a process report, use any of the following:

- standard shell command for viewing a text file
- Performance Metrics Service
- `VIEW REPORT` command.
- To view information if a process abends without generating a report, use the following command to run the process from the command shell of the operating system (not Oracle GoldenGate command line) to send the information to the terminal.

```
process paramfile path.prm
```

Where:

- The value for `process` is either `extract` or `replicat`.
- The value for `path.prm` is the fully qualified name of the parameter file, for example:

```
REPLICA PARAMFILE /ogg/dirdat/repora.prm
```

By default, reports have a file extension of `.rpt`, for example `EXTORA.rpt`. The default location is the `dirrpt` sub-directory of the Oracle GoldenGate directory. However, these properties can be changed when the group is created. Once created, a report file must remain in its original location for Oracle GoldenGate to operate properly after processing has started.

To determine the name and location of a process report, use the `INFO EXTRACT`, or `INFO REPLICAT` commands.

Scheduling Runtime Statistics in the Process Report

By default, runtime statistics are written to the report once, at the end of each run. For long or continuous runs, you can use optional parameters to view these statistics on a regular basis, without waiting for the end of the run.

To set a schedule for reporting runtime statistics, use the `REPORT` parameter in the Extract or Replicat parameter file to specify a day and time to generate runtime statistics in the report. See `REPORT`.

To send runtime statistics to the report on demand, use the `SEND EXTRACT` or `SEND REPLICAT` command with the `REPORT` option to view current runtime statistics when needed.

Viewing Record Counts in the Process Report

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.

Prevent SQL Errors from Filling the Replicat Report File

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.

Use the Discard File

By default, a discard file is generated whenever a process is started with the `START` command. The discard file captures information about Oracle GoldenGate operations that failed. This information can help you resolve data errors, such as those that involve invalid column mapping.

The discard file reports such information as:

- The database error message
- The sequence number of the data source or trail file
- The relative byte address of the record in the data source or trail file
- The details of the discarded operation, such as column values of a DML statement or the text of a DDL statement.

To view the discard file, use a text editor or use the `VIEW REPORT` command in Admin Client.

The default discard file has the following properties:

- The file is named after the process that creates it, with a default extension of `.dsc`.
Example: `finance.dsc`.
- The file is created in the `dirrpt` sub-directory of the Oracle GoldenGate installation directory.
- The maximum file size is 50 megabytes.
- At startup, if a discard file exists, it is purged before new data is written.

You can change these properties by using the `DISCARDFILE` parameter. You can disable the use of a discard file by using the `NODISCARDFILE` parameter.

If a process is started from the command line of the operating system, it does not generate a discard file by default. You can use the `DISCARDFILE` parameter to specify the use of a discard file and its properties.

Once created, a discard file must remain in its original location for Oracle GoldenGate to operate properly after processing has started.

Maintain Discard and Report Files

By default, discard files and report files are aged the same way. A new discard or report file is created at the start of a new process run. Old files are aged by appending a sequence number from 0 (the most recent) to 9 (the oldest) to their names.

If the active report or discard file reaches its maximum file size before the end of a run (or over a continuous run), the process abends unless there is an aging schedule in effect. Use the `DISCARDROLLOVER` and `REPORTROLLOVER` parameters to set aging schedules for the discard and report files respectively. These parameters set instructions for rolling over the files at regular intervals, in addition to when the process starts. Not only does this control the size of the files and prevent process outages, but it also provides a predictable set of archives that can be included in your archiving routine. For more information, see the following documentation:

- `DISCARDROLLOVER`
- `REPORTROLLOVER`

No process ever has more than ten aged reports or discard files and one active report or discard file. After the tenth aged file, the oldest is deleted when a new report is created. It is recommended that you establish an archiving schedule for aged reports and discard files in case they are needed to resolve a service request.

Table 10-1 Current Extract and Aged Reports

Permissions	X	Date	Report
-rw-rw-rw-	1 ggs ggs	4384 Oct 5 14:02	TCUST.rpt
-rw-rw-rw-	1 ggs ggs	1011 Sep 27 14:10	TCUST0.rpt
-rw-rw-rw-	1 ggs ggs	3184 Sep 27 14:10	TCUST1.rpt
-rw-rw-rw-	1 ggs ggs	2655 Sep 27 14:06	TCUST2.rpt
-rw-rw-rw-	1 ggs ggs	2655 Sep 27 14:04	TCUST3.rpt
-rw-rw-rw-	1 ggs ggs	2744 Sep 27 13:56	TCUST4.rpt
-rw-rw-rw-	1 ggs ggs	3571 Aug 29 14:27	TCUST5.rpt

Parameters Used to Interpret Synchronization Lag

The time differences between source and target systems is known as the synchronization lag. To account for this lag, use the `TCPSOURCE_TIMER | NOTCPSOURCE_TIMER` parameter in the Extract parameter file. This parameter adjusts the timestamps of replicated records for reporting purposes, making it easier to interpret synchronization lag.

Tuning

Learn about tuning the performance of Oracle GoldenGate.

Topics:

Tuning the Performance of Oracle GoldenGate

See Tuning the Performance of Oracle GoldenGate in the *Administering Oracle GoldenGate* guide.

Autonomous Database

This section provides details about configuring Oracle GoldenGate with Oracle Autonomous Database, and using Extract and Replicat processes with Autonomous Database instances.

About Capturing and Replicating Data Using Autonomous Databases

You can capture changes from the Oracle Autonomous Database instance and replicate to any **target database** or platform that Oracle GoldenGate supports, including another Oracle Autonomous Database instance.

Use Case: When Using Oracle GoldenGate with Autonomous Databases

Using Oracle GoldenGate in the Oracle Autonomous Database can be configured to support the following scenarios:

- **Scalable Active-Active architecture:** Synchronize changes made across two or more databases to scale out workloads, provide increase resilience and near instantaneous failover across multiple data centers or regions.
- **Real-Time Data Warehouse:** Provide continuous, real-time capture and delivery of changed data between Oracle Autonomous Database systems.
- **Big Data Integration:** With Oracle GoldenGate for Big Data you can replicate data from the Oracle Autonomous Database to provide real-time streaming integration to all platforms supported by Big Data targets.
- **Real-Time Streaming Analytics:** Oracle GoldenGate integrates seamlessly with Oracle Stream Analytics to enable users to identify events of interest by executing queries against event streams in real time. It allows creating custom operational dashboards that provide real-time monitoring, transform streaming data, or raise alerts based on stream analysis.
- **Hybrid Replication:** Oracle GoldenGate replicates data from the Oracle Autonomous Database instance back to on-premise or to another cloud database or platform.

The following features are not available with Always Free Autonomous Databases:

- Supplemental logging
- Oracle GoldenGate Extract

See Always Free Autonomous Database for details.

Details of Support When Using Oracle GoldenGate with Autonomous Databases

Review the supported data types and limitations before replicating data to the Oracle Autonomous Database instance.

Oracle GoldenGate is supported for any type of Oracle Autonomous Database.

Details of Support for coexistence of Oracle GoldenGate with Transient Logical Rolling Upgrades

Coexistence of Oracle GoldenGate Extract and Replicat processes with Transient Logical Rolling Upgrades is supported.

Limitations of Extract and Replicat during Transient Logical Rolling Upgrades

- Creation of new Extracts/Replicats are not supported during rolling upgrade.
- Existing Extracts/Replicats will continue to capture or apply changes. However, there are restrictions on modifying the Extracts/Replicats to points before and after a rolling upgrade. You cannot alter Extract to a point before rolling upgrade after switching to the new primary.

Oracle GoldenGate Replicat Limitations for Autonomous Databases

These are the limitations of Oracle GoldenGate when replicating to or from the Oracle Autonomous Database.

Supported Replicats

The following combinations of Replicats are supported in different modes when using Oracle GoldenGate with Oracle Autonomous Database:

- Parallel Replicat in integrated mode is supported for Oracle Autonomous Database Serverless.
- Classic and coordinated Replicats in integrated mode are not supported for Oracle Autonomous Database.
- Classic, coordinated, and parallel Replicats in non-integrated mode are supported for Oracle Autonomous Database.

Data Type Limitations for DDL and DML Replication

See the section [Non-Supported Oracle Data Types](#).

Also see Data Types in the *Autonomous Database on Dedicated Exadata Infrastructure Documentation* and Data Types in the *Using Oracle Autonomous Database Serverless* guide. DDL replication is supported depending on the restrictions in the Autonomous Databases.

Details of Support for Archived Log Retention

The two types of Autonomous Databases, Oracle Autonomous Database Serverless and Oracle Autonomous Database on Dedicated Exadata Infrastructure have different log retention behavior.

- Oracle Autonomous Database Serverless: Archived log files are kept in Fast Recovery Area (FRA) for up to 48 hours. After that, it is purged and the archived log files are moved to NFS mount storage, which is accessible by logminer. Three copies are created. The logminer should be able to access any of the copies. This is transparent to Oracle GoldenGate Extract. After it reaches 7 days, the NFS mounted copy is permanently removed. The Extract abends with the `archived log unavailable` error if the required archived log file is older than 7 days.
- Oracle Autonomous Database on Dedicated Exadata Infrastructure: When Oracle Autonomous Data Guard or Oracle GoldenGate is enabled, archived log files are kept in Fast Recovery Area (FRA) for up to 7 days. After that, the files are purged. There is no NFS mount location available for logminer to access archived log files that are older than 7 days. The Extract abends with the `archived log unavailable` error if the required archived log file is older than 7 days.

 **Note:**

If the database instance is closed for more than 15 minutes, then the retention time is set back to 3 days. This implies that retention of archived log files is confirmed only for 3 days, regardless of whether the database instance is closed. The files are retained for 7 days only if the database instance is not closed.

Configure Extract to Capture from an Autonomous Database

Oracle Autonomous Database has a tight integration with Oracle GoldenGate. There are a number of differences when setting up Extract for an Autonomous database instance compared to a traditional Oracle Database.

Oracle Autonomous Database security has been enhanced to ensure that Extract is only able to capture changes from the specific tenant it connected to. However, downstream Extract is not supported.

Before You Begin

Before you start the process of capturing data from the Autonomous Database using Oracle GoldenGate you must first:

1. Unlock the pre-created Oracle GoldenGate database user `ggadmin` in the Autonomous Database.
2. Obtain the Autonomous Database client credentials to connect to the database instance.

Establishing Oracle GoldenGate Credentials

To capture from an Autonomous Database only the `GGADMIN` account is used. The `GGADMIN` account is created inside the database when the Autonomous Database is provisioned. This account is locked. It must be unlocked before it can be used with Oracle GoldenGate. This account is the same account used for both Extracts and Replicats in the Autonomous Database.

Run the `ALTER USER` command to unlock the `ggadmin` user and set the password for it. See [Creating Users with Autonomous Database with Client-Side Tools](#).

This `ALTER USER` command must be run by the `admin` account user for Autonomous Databases.

```
ALTER USER ggadmin IDENTIFIED BY PASSWORD ACCOUNT UNLOCK;
```

Prerequisites for Configuring Oracle GoldenGate Extract to Capture from Autonomous Databases

Prior to configuring and starting the Extract process to capture from the Autonomous Database, make sure that the following requirements are met:

- Oracle Autonomous Database environment is provisioned and running.

- Autonomous Database-level supplemental logging should be enabled by the `ADMIN` or `GGADMIN`.

Configuring Autonomous Database Supplemental Logging for Extract

To add minimal supplemental logging to your Autonomous Database instance, log into the instance as `GGADMIN` or `ADMIN` account and execute the following commands:

```
ALTER PLUGGABLE DATABASE ADD SUPPLEMENTAL LOG DATA;
```

To `DROP` Autonomous Database-level supplemental logging incase you decide to stop capturing from that database instance:

```
ALTER PLUGGABLE DATABASE DROP SUPPLEMENTAL  
LOG DATA;
```

You can verify that the Autonomous Database-level supplemental logging is configured properly by issuing this SQL statement:

```
SELECT MINIMAL FROM dba_supplemental_logging;
```

The output for this statement is:

```
MINIMAL  
-----  
YES
```

The `MINIMAL` column will be `YES` if supplemental logging has been correctly set for this Autonomous Database instance.

Configure Extract to Capture from an Autonomous Database

Following are the steps to configure an Extract to capture from an Oracle Autonomous Database :

1. Install Oracle GoldenGate for your Oracle Autonomous Database instance.
2. Create a deployment for the Oracle GoldenGate environment. This is the deployment where the Extract that captures data from the Oracle Autonomous Database instance will be created. See [Add a Deployment](#).
3. Obtain Oracle Autonomous Database Client Credentials.

To establish connection to your Oracle Autonomous Database instance, download the client credentials file. To download client credentials, you can use the Oracle Cloud Infrastructure Console or Database Actions Launchpad. See [Downloading Client Credentials \(Wallets\)](#).

Note:

If you do not have administrator access to the Oracle Autonomous Database, you should ask your service administrator to download and provide the credentials files to you.

The following steps use the **Database Actions Launchpad** to download the client credentials.

- a. Log in to your Oracle Autonomous Database account.
- b. From the **Database Instance** page, click **Database Actions**. This launches the Database Actions Launchpad. The Launchpad attempts to log you into the database as ADMIN. If that is not successful, you will be prompted for your database ADMIN username and password.
- c. On the **Database Actions** Launchpad, under **Administration**, click **Download Client Credentials (Wallets)**.
- d. Enter a password to secure your Client Credentials zip file and click **Download**.

 **Note:**

The password you provide when you download the wallet protects the downloaded Client Credentials wallet.

- e. Save the credentials zip file to your local system.

The credentials zip file contains the following files:

- cwallet.sso
- ewallet.p12
- keystore.jks
- ojdbc.properties
- sqlnet.ora
- tnsnames.ora
- truststore.jks
- ewallet.pem
- README.txt

Refer and update (if required) the `sqlnet.ora` and `tnsnames.ora` files while configuring Oracle GoldenGate to work with the Autonomous Database instance.

4. Configure the server where Oracle GoldenGate is running to connect to the Autonomous Database instance.
 - a. Log in to the server where Oracle GoldenGate was installed.
 - b. Transfer the credentials zip file that you downloaded from Oracle Autonomous database instance to the Oracle GoldenGate server.
 - c. In the Oracle GoldenGate server, unzip the credentials file into a new directory, for example: `/u02/data/adwc_credentials`. This is your key directory.
 - d. To configure the connection details, open your `tnsnames.ora` file from the Oracle client location in the Oracle GoldenGate instance.
 - e. Use the connection string with the LOW consumer group `dbname_low`, for example, `graphdb1_low`, and move it to your local `tnsnames.ora` file.

See *Local Naming Parameters in the tnsnames.ora File* chapter in the *Oracle Database Net Services Reference guide*.

 **Note:**

The `tnsnames.ora` file provided with the credentials file contains three database service names identifiable as:

```
ADWC_Database_Name_low
ADWC_Database_Name_medium
ADWC_Database_Name_high
```

Oracle recommends that you use `ADWC_Database_Name_low` with Oracle GoldenGate. See [Predefined Database Service Names for Autonomous Database](#) in the *Using Oracle Autonomous Database Serverless* guide or [Predefined Database Service Names for Autonomous Databases](#) for Oracle Autonomous Database on Dedicated Exadata Infrastructure.

- f. Edit the `tnsnames.ora` file in the Oracle GoldenGate instance to include the connection details available in the `tnsnames.ora` file in your key directory (the directory where you unzipped the credentials zip file downloaded from the Autonomous Database).

Sample Connection String

```
adwl_low. = (description=
              (retry_count=20) (retry_delay=3)
              (address=(protocol=tcps) (port=1522) (host=adb-
preprod.us-phoenix-1.oraclecloud.com) )

              (connect_data=(service_name=okd2ybgcz4mjx94_graphdb1_low.adb.oraclecloud
.com) )
              (security=(ssl_server_cert_dn="CN=adwc-preprod.uscom-
east-1.oraclecloud.com,OU=Oracle BMCS US,O=Oracle Corporation,L=Redwood
City,ST=California,C=US"))
              )
```

If the database is within a firewall protected environment, you might not have direct access to the database. With an existing HTTP Proxy, you can pass the firewall with the following modifications to the `sqlnet.ora` and `tnsnames.ora`:

- *sqlnet parameters*
- *address modification of tns_alias*

If Extract becomes unresponsive due to a network timeout or connection loss, then you can add the following into the connection profile in the `tnsnames.ora` file:

```
(DESCRIPTION = (RECV_TIMEOUT=30) (ADDRESS_LIST =
                (LOAD_BALANCE=off) (FAILOVER=on) (CONNECT_TIMEOUT=3) (RETRY_COUNT=3)
                (ADDRESS = (PROTOCOL = TCP) (HOST = adb-preprod.us-
phoenix-1.oraclecloud.com) (PORT = 1522))
```

- g. To configure the wallet, create a `sqlnet.ora` file in the Oracle client location in the Oracle GoldenGate instance.

```
cd /u02/data/oci/network/admin
ls
sqlnet.ora tnsnames.ora
```

See Autonomous Database Client Credentials in *Using Oracle GoldenGate on Oracle Cloud Marketplace*.

- h. Edit this `sqlnet.ora` file to include your key directory.

```
WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA =
(DIRECTORY="/u02/data/adwc_credentials")))
SSL_SERVER_DN_MATCH=yes
```

5. Use Admin Client to log into the Oracle GoldenGate deployment, depending on whether you are using Microservices.
6. Create a credential to store the `GGADMIN` user and password. This user will be used to connect to the Autonomous Database from the command line, to perform commands that require a database connection. It will also be used in the `USERIDALIAS` parameter for the Extract database connection.

```
ALTER CREDENTIALSTORE ADD USER
ggadmin@dbgraph1_low PASSWORD complex_password alias adb_alias
```

7. Connect to the database using `DBLOGIN`. The `DBLOGIN` user should be the `adb_alias` account user.

```
DBLOGIN USERIDALIAS adb_alias
```

8. Configure supplemental logging on the tables, which you want to capture using `ADD TRANDATA` or `ADD SCHEMATRANDATA`. Remember that you are connected directly to the database instance, so there is no need to include the database name in these commands. Here's an example:

```
ADD TRANDATA HR.EMP
```

or

```
ADD SCHEMATRANDATA HR
```

See [Prerequisites for Configuring Oracle GoldenGate Extract to Capture from Autonomous Databases](#).

9. Add heartbeat table.

```
ADD HEARTBEATTABLE
```

10. Add and configure an Extract to capture from the Oracle Autonomous Database instance. See [Add an Extract](#) for steps to create an Extract.

Oracle GoldenGate Extract is designed to work with the Oracle Autonomous Database instance to ensure that it only captures from a specific database instance. This means that the database instance name is not needed for any `TABLE` or `MAP` statements.

The following example creates an Extract (required for capturing from an Oracle Autonomous Database) called `exte`, and instructs it to begin now.

```
ADD EXTRACT exte, INTEGRATED TRANLOG, BEGIN NOW
```

To capture specific tables, use the two part object names.. For example, to capture from the table `HR.EMP`, in your Oracle Autonomous Database instance, use this entry in the Extract parameter file.

```
TABLE HR.EMP;
```

If you want to replicate `HR.EMP` into `COUNTRY.EMPLOYEE`, then your map statement would look like this:

```
MAP HR.EMP, TARGET COUNTRY.EMPLOYEE;
```

11. Register Extract with the Oracle Autonomous Database instance. For example, to register an Extract named `exte`, use the following command:

```
REGISTER EXTRACT exte DATABASE
```

12. You can now start your Extract and perform data replication to the Oracle Autonomous Database instance. Here's an example:

```
START EXTRACT exte
```

This completes the process of configuring an Extract for Oracle Autonomous Database and you can use it like any other Extract process.

Configure Replicat to Apply to an Oracle Autonomous Database

You can replicate into the Autonomous Database from any source database or platform that is supported by Oracle GoldenGate.

Prerequisites for Configuring Oracle GoldenGate Replicat to an Autonomous Database

You should have the following details available with you:

- Your source database with Oracle GoldenGate Extract processes configured and writing trails to where the Replicat is running to apply data to the Autonomous Database target.
- Oracle Autonomous Database is environment provisioned and running.

To deliver data to the Autonomous Database instance using Oracle GoldenGate, perform the following tasks:

Configure Oracle GoldenGate for an Autonomous Database

Learn the steps to configure Oracle GoldenGate Replicat for an Autonomous Database.

Here are the steps to complete the configuration tasks:



Note:

Instructions are based on the assumption that the source environment is already configured.

1. For Oracle GoldenGate on-premises, make sure that Oracle GoldenGate is installed.
2. (Microservices only) Create a deployment for your Oracle GoldenGate environment. This is the deployment where the Replicat that applies data into the Autonomous Database (ADB) will be created. See [How to Create Deployments](#) for steps to add a deployment.
3. The Autonomous Database has a pre-existing user created for Oracle GoldenGate on-premise called `ggadmin`. The `ggadmin` user has been granted the required privileges for Replicat to work. This is the user where any objects used for Oracle GoldenGate processing will be stored, like the checkpoint table and heartbeat objects. By default, this user is locked. To unlock the `ggadmin` user, connect to the Oracle Autonomous Database instance as the `ADMIN` user using any SQL client tool. See [Create Users on Autonomous Database with Database Actions](#).
4. Run the `ALTER USER` command to unlock the `ggadmin` user and set the password for it. This will be used in GGSCI or Admin Client for any `DBLOGIN` operations on the Autonomous Database. It will be used in Replicat to allow Oracle GoldenGate to connect to the Autonomous Database and apply data. See [Create Users on Autonomous Database with Database Actions](#).

```
ALTER USER ggadmin IDENTIFIED BY p0$$word ACCOUNT UNLOCK;
```

Obtain the Autonomous Database Client Credentials

To establish a connection with an Oracle Autonomous Database instance, you need to download the client credentials files. There are two ways to download the client credentials files: the Oracle Cloud Infrastructure Console or Database Actions Launchpad.

For details, see [Downloading Client Credentials \(Wallets\)](#).



Note:

If you do not have administrator access to the Oracle Autonomous Database, you should ask your service administrator to download and provide the credentials files to you.

The following steps use the **Database Actions Launchpad** to download the client credentials files.

1. Log in to your Autonomous Database account.

2. From the **Database Instance** page, click **Database Actions**. This launches the Database Actions Launchpad. The Launchpad attempts to log you into the database as ADMIN. If that is not successful, you will be prompted for your database ADMIN username and password.
3. On the **Database Actions** Launchpad, under **Administration**, click **Download Client Credentials (Wallets)**.
4. Enter a password to secure your Client Credentials zip file and click **Download**.

 **Note:**

The password you provide when you download the wallet protects the downloaded Client Credentials wallet.

5. Save the credentials zip file to your local system. The credentials zip file contains the following files:
 - cwallet.sso
 - ewallet.p12
 - keystore.jks
 - ojdbc.properties
 - sqlnet.ora
 - tnsnames.ora
 - truststore.jks
 - ewallet.pem
 - README.txt

Refer and update (if required) the `sqlnet.ora` and `tnsnames.ora` files while configuring Oracle GoldenGate to work with the Oracle Autonomous Database instance.

Configure Replicat to Apply to an Autonomous Database

This section assumes that the source environment is already configured and provides the steps required to establish replication in the Oracle Autonomous Database environment.

In the Oracle GoldenGate instance, you need to complete the following:

1. Follow the steps given in [Prerequisites for Configuring Oracle GoldenGate Replicat to an Autonomous Database](#).
2. Follow the steps given in [Configure Oracle GoldenGate for an Autonomous Database](#).
3. Follow the steps given in [Obtain the Autonomous Database Client Credentials](#).
4. Log in to the server where Oracle GoldenGate was installed.
5. Transfer the credentials zip file that you downloaded from Oracle Autonomous Database to your Oracle GoldenGate instance.
6. In the Oracle GoldenGate instance, unzip the credentials file into a new directory `/u02/data/adwc_credentials`. This is your key directory.

7. To configure the connection details, open your `tnsnames.ora` file from the Oracle client location in the Oracle GoldenGate instance.

```
cd /u02/data/adwc_credentials
ls
tnsnames.ora
```

8. Edit the `tnsnames.ora` file in the Oracle GoldenGate instance to include the connection details available in the `tnsnames.ora` file in your key directory (the directory where you unzipped the credentials zip file downloaded from Oracle Autonomous Database).

Sample Connection String

```
graphdb1_low = (description=
                (retry_count=20) (retry_delay=3) (address=(protocol=tcps)
                (port=1522) (host=adb-preprod.us-phoenix-1.oraclecloud.com))

(connect_data=(service_name=okd2ybgcz4mjx94_graphdb1_low.adb.oraclecloud.co
m))

                (security=(ssl_server_cert_dn="CN=adwc-preprod.uscom-
east-1.oraclecloud.com,OU=Oracle BMCS US,O=Oracle Corporation,L=Redwood
City,ST=California,C=US")))
```

If Replicat becomes unresponsive due to a network timeout or connection lost, then you can add the following into the connection profile in the `tnsnames.ora` file:

```
(DESCRIPTION = (RECV_TIMEOUT=120) (ADDRESS_LIST =
                (LOAD_BALANCE=off) (FAILOVER=on) (CONNECT_TIMEOUT=3) (RETRY_COUNT=3)
                (ADDRESS = (PROTOCOL = TCP) (HOST = adb-preprod.us-
phoenix-1.oraclecloud.com) (PORT = 1522))
```

 **Note:**

The `tnsnames.ora` file provided with the credentials file contains three database service names identifiable as:

```
ADWC_Database_Name_low
ADWC_Database_Name_medium
ADWC_Database_Name_high
```

For Oracle GoldenGate replication, use `ADWC_Database_Name_low`.

9. To configure the wallet, create a `sqlnet.ora` file in the Oracle client location in the Oracle GoldenGate instance.

```
cd /u02/data/oci/network/admin
ls
sqlnet.ora tnsnames.ora
```

10. Edit this `sqlnet.ora` file to include your key directory.

```
WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA =  
(DIRECTORY="/u02/data/adwc_credentials")))  
SSL_SERVER_DN_MATCH=yes
```

11. Use the Admin Client to log in to the Oracle GoldenGate deployment.
12. Create a credential to store the `GGADMIN` user and password for the Replicat to use. For example:

```
ADD CREDENTIALSTORE ALTER CREDENTIALSTORE ADD USER  
ggadmin@databasename_low PASSWORD complex_password alias adb_alias
```

13. Add and configure a Replicat to deliver to Oracle Autonomous Database. When creating the Replicat, use the alias created in the previous step. For setting up your Replicat and other processes, see [Add a Replicat](#).

 **Note:**

You can use classic Replicat, coordinated Replicat, and parallel Replicat in non-integrated mode. Parallel Replicat in integrated mode is also supported for Oracle Autonomous Database.

14. You can now start your Replicat and perform data replication to the Autonomous Database.

 **Note:**

Oracle Autonomous Database times out and disconnects the Replicat when it is idle for more than 60 minutes. When Replicat tries to apply changes (when it gets new changes) after being idle, it encounters a database error and abends. Oracle recommends that you configure Oracle GoldenGate with the `AUTORESTART` profile using managed processes (Microservices Architecture) to avoid having to manually restart a Replicat when it times out.

12

Upgrade

Learn about the tasks required for upgrading Oracle GoldenGate Microservices Architecture.

Obtaining the Oracle GoldenGate Distribution

To obtain Oracle GoldenGate, follow these steps:

1. Go to edelivery: edelivery.oracle.com

Also see [Oracle GoldenGate -- Oracle RDBMS Server Recommended Patches \(Doc ID 1557031.1\)](#) for more information.

To access Oracle Technology Network, go to <https://www.oracle.com/middleware/technologies/goldengate.html>

2. Find the Oracle GoldenGate 19c (19.1.0) release and download the ZIP file onto your system.

For more information about locating and downloading Oracle Fusion Middleware products, see the [Oracle Fusion Middleware Download, Installation, and Configuration Readme Files](#) on Oracle Technology Network.

Prerequisites

Learn about the prerequisites of performing an upgrade for Oracle GoldenGate.

For Microservices, the earliest version that can be upgraded from is Oracle GoldenGate 12c (12.3.0.1). As a best practice, perform a minimal upgrade first, so that you can troubleshoot more easily in the event that any problems arise. Once you know your environment is upgraded successfully, you can implement the new functionality.

The upgrade instructions also include the steps for upgrading the source or target database and Oracle GoldenGate at the same time. Following are the pre-upgrade requirements:

- Stop all Oracle GoldenGate processes.
- Upgrade the database and then start it.
- Start Oracle GoldenGate.
- Disable the DDL trigger if there is no native DDL support.

Oracle GoldenGate Upgrade Considerations

Before you start the upgrade, review the information about upgrading Extract and Replicat.

Even though you may only be upgrading the source or target, rather than both, all processes are involved in the upgrade. All processes must be stopped in the correct order for the upgrade, regardless of which component you upgrade, and the trails must be processed until empty.

Oracle recommends that you begin your upgrade with the target rather than the source to avoid the necessity of adjusting the trail file format.

Extract Upgrade Considerations

If you are using trigger-based DDL support, you must rebuild the DDL objects, even if you plan to use the new triggerless DDL support in an integrated capture. After the upgrade, when Oracle GoldenGate is running successfully again, you can remove the trigger and DDL objects.

The output trail file is automatically rolled over when the Extract restarts and the integrated Extract version is upgraded.

Because the time zone is different, you may need to run the `ALTER REPLICAT extseqno` command to synchronize with newer trail files after consuming the old trail file written by Integrated Extract version 1.

After completing the upgrade, run the `UPGRADE HEARTBEATABLE` command to add extra columns for tables and lag views. These extra columns are used to track the Extract restart position. See `UPGRADE HEARTBEATABLE` to know more.

Replicat Upgrade Considerations

All Replicat installations should be upgraded at the same time. It is critical to ensure that all trails leading to all Replicat groups on all target systems are processed until empty, according to the upgrade instructions.

Caution:

The hash calculation used by the `@RANGE` function to partition data among Replicat processes has been changed. This change is transparent, and no re-partitioning of rows in the parameter files is required, so long as the upgrade is performed as directed in these instructions. To ensure data continuity, make certain to allow all Replicat processes on all systems to finish processing all of the data in their trails before stopping those processes, according to the upgrade instructions. Note that if the Replicat processes are not upgraded all at the same time, or the trails are not cleaned out prior to the upgrade, rows may shift partitions as a result of the new hash method, which may result in collision errors.

When upgrading from the 18c release of Oracle GoldenGate to the 19c (19.1.0) release, ensure that you do not use the `SOURCEDEF` parameter in Replicat, otherwise the Replicat will abend. However, if the trail file format is pre-12.2, then `SOURCEDEF` is still required because no metadata exists in the trail file.

For PostgreSQL: When upgrading from 12.2.0.1 to 19c (19.1.0), you need to run the `UPGRADE HEARTBEATABLE` command from GGSCI to upgrade the heartbeatable to include the `LOGBSN` columns

Upgrading Oracle GoldenGate Microservices – GUI Based

Learn the steps to upgrade Oracle GoldenGate Microservices using the GUI.

Follow these steps to obtain the Oracle GoldenGate installation software and set up the directories for upgrade. Now, perform the following steps:

1. Verify the current version of Oracle GoldenGate Home through Service Manager.
 - a. Login to the Service Manager: `http://host:servicemanager_port`
 - b. Review the deployment section for your current Oracle GoldenGate home location.
2. Update the Service Manager and the deployments with the location of the new Oracle GoldenGate home.
 - a. Click **Service Manager**, then the **Deployment name** link.
 - b. Next to the deployment details, click the pencil icon. This opens the dialog box to edit the Oracle GoldenGate home.
 - c. Update the Oracle GoldenGate home with the complete path to the new Oracle GoldenGate home. Also update the `LD_LIBRARY_PATH`, if required.
 - d. Click **Apply**.
 - e. Confirm that the Oracle GoldenGate home has been updated.
 - f. Stop all Extracts, Replicats, and Distribution paths.
 - g. Use the action button to restart **Service Manager** or **Deployment**.

 **Note:**

You can confirm that the Oracle GoldenGate home was updated by looking at the process from the operating system for Service Manager. The Service Manager process should be running from the new Oracle GoldenGate home.

3. To upgrade the associated deployments, follow the same steps for Service Manager after ensuring that all the Extract and Replicat processes in that deployment have been stopped.

Upgrading Oracle GoldenGate Microservices Using REST APIs

Learn how to upgrade Oracle GoldenGate MA to Oracle GoldenGate MA 19c (19.1.0) using REST APIs.

Follow these steps to perform the upgrade using REST APIs:

1. Download the latest Oracle GoldenGate MA 19c software from the Oracle Technology Network or eDelivery.
2. Upload the Oracle GoldenGate MA 19c software to a staging location on the server where a previous release of Oracle GoldenGate Microservices exists.
3. Unzip Oracle GoldenGate MA 19c software in the staging location.

```
$ cd /tmp
```

```
$ unzip ./fbo_ggs_Linux_x64_services_shiphome.zip
```

4. Untar the tar file that gets created after the unzip command:

```
tar -xvf ggs_Linux_x64_Oracle_64bit.tar
```

5. Move into the unzipped files and execute the `runInstaller` command.

```
$ cd ./fbo_ggs_Linux_x64_services_shiphome/Disk1
```

```
$ ./runInstaller
```

6. For **Software Location**, specify where the new Oracle GoldenGate home will be located. This will not be the same location as the current Oracle GoldenGate home. Click **Next**.
7. Click **Install** to begin installing the new Oracle GoldenGate MA. When the installation is done, click **Close**.
8. At this point, you should have two Oracle GoldenGate MA home directories: one for your old home (12c or 18c) and a new home (19c).

Now, you are ready to update the Oracle GoldenGate MA home (OGG_HOME) for the Service Manager or deployments using REST API.

Upgrade a Service Manager

To upgrade the Service Manager, the following cURL command can be used to update the Oracle GoldenGate home:

```
curl -X PATCH \
  https://<hostname>:<port>/services/v2/deployments/ServiceManager \
  -H 'cache-control: no-cache' \
  -d '{"oggHome":"/opt/oracle/product/19.1.0/oggcore_1", "status":"restart"}'
```

Upgrade a Deployment

To upgrade a Deployment:

1. Stop all Extract and Replicats within the Administration Service.
2. Stop all Distribution Paths within the Distribution Service.
3. Run this simple cURL command to update the Oracle GoldenGate home:

```
curl -X PATCH \
  https:// ://<hostname>:<port>/services/v2/deployments/<deployment name> \
  -H 'cache-control: no-cache' \
  -d '{"oggHome":"/opt/app/oracle/product/19.1.0/oggcore_1",
    "status":"restart"}'
```

Note:

You can confirm that the Oracle GoldenGate home was updated by looking at the process from the operating system for Service Manager. The Service Manager process should be running from the new Oracle GoldenGate home.

4. Start all distribution paths within the Distribution Server.
5. Start all Extracts and Replicats within the Administration Server.

Once the Service Manager or Deployment restarts, you are upgraded to the next version.

Appendix

Learn about additional details required for supporting Oracle GoldenGate on different databases.

Using the LogDump Utility to Access Trail File Records

Oracle GoldenGate trail information is required for troubleshooting and technical support. Use the Logdump utility to view the Oracle GoldenGate trail records.

Trail Recovery Mode

By default, Extract operates in append mode, where if there is a process failure, a recovery marker is written to the trail and Extract appends recovery data to the file so that a history of all prior data is retained for recovery purposes.

In append mode, the Extract initialization determines the identity of the last complete transaction that was written to the trail at startup time. With that information, Extract ends recovery when the commit record for that transaction is encountered in the data source; then it begins new data capture with the next committed transaction that qualifies for extraction and begins appending the new data to the trail. A Replicat starts reading again from that recovery point.

Overwrite mode is another version of Extract recovery that was used in versions of Oracle GoldenGate prior to version 10.0. In these versions, Extract overwrites the existing transaction data in the trail after the last write-checkpoint position, instead of appending the new data. The first transaction that is written is the first one that qualifies for extraction after the last read checkpoint position in the data source.

If the version of Oracle GoldenGate on the target is older than version 10, Extract will automatically revert to overwrite mode to support backward compatibility. This behavior can be controlled manually with the `RECOVERYOPTIONS` parameter.

Trail Record Format

Each change record written by Oracle GoldenGate to a trail or Extract file includes a header area, a data area, and possibly a user token area. The record header contains information about the transaction environment, and the data area contains the actual data values that were extracted.

The token area contains information that is specified by Oracle GoldenGate users for use in column mapping and conversion.

Oracle GoldenGate trail files are unstructured. You can view Oracle GoldenGate records with the Logdump utility provided with the Oracle GoldenGate software. For more information, see *Viewing the First Record in the Logdump Reference for Oracle GoldenGate*.

**Note:**

As enhancements are made to the Oracle GoldenGate software, the trail record format is subject to changes that may not be reflected in this documentation. To view the current structure, use the Logdump utility.

Trail File Header Record

Each file of a trail contains a file header record that is stored at the beginning of the file. The file header contains information about the trail file itself. Previous versions of Oracle GoldenGate do not contain this header.

The file header is stored as a record at the beginning of a trail file preceding the data records. The information that is stored in the trail header provides enough information about the records to enable an Oracle GoldenGate process to determine whether the records are in a format that the current version of Oracle GoldenGate supports.

The trail header fields are stored as tokens, where the token format remains the same across all versions of Oracle GoldenGate. If a version of Oracle GoldenGate does not support any given token, that token is ignored. Depreciated tokens are assigned a default value to preserve compatibility with previous versions of Oracle GoldenGate.

To ensure forward and backward compatibility of files among different Oracle GoldenGate process versions, the file header fields are written in a standardized token format. New tokens that are created by new versions of a process can be ignored by older versions, so that backward compatibility is maintained. Likewise, newer Oracle GoldenGate versions support older tokens. Additionally, if a token is deprecated by a new process version, a default value is assigned to the token so that older versions can still function properly. The token that specifies the file version is `COMPATIBILITY` and can be viewed in the Logdump utility and also by retrieving it with the `GGFILEHEADER` option of the `@GETENV` function.

A trail or Extract file must have a version that is equal to, or lower than, that of the process that reads it. Otherwise the process will abend. Additionally, Oracle GoldenGate forces the output trail to be the same version as that of its input trail or file. Upon restart, Extract rolls a trail to a new file to ensure that each file is of only one version (unless the file is empty).

From Oracle GoldenGate 21c onward, for Oracle databases, you can specify a globally unique name for the database using the `DB_UNIQUE_NAME` parameter. If this database parameter is not set, then the `DB_UNIQUE_NAME` is the same as `DB_NAME`. This feature allows unique identification of the source of the trail data by viewing the trail file header.

See [GETENV](#) parameter to know about the use of the `DbUniqueName` token.

The `DbUniqueName` token will be written to trail files with 19.1 compatibility level, however prior Oracle GoldenGate releases supporting that compatibility level will ignore the new token. The token belongs to the Database Information group. The field will be limited to 65536 bytes, to allow fitting all possible values of `DB_UNIQUE_NAME`, limited to 30 characters.

Because the Oracle GoldenGate processes are decoupled and can be of different Oracle GoldenGate versions, the file header of each trail file contains a version indicator. By default, the version of a trail file is the current version of the process that created the file. If you need to set the version of a trail, use the `FORMAT` option of the `EXTTRAIL`, `EXTFILE`, `RMTRAIL`, or `RMTFILE` parameter.

You can view the trail header with the `FILEHEADER` command in the Logdump utility. For more information about the tokens in the file header, see [Logdump Reference for Oracle GoldenGate](#).

Partition Name Record in Trail File Header

Each DML record in the trail file header can contain an index to a partition name record (PNR). Because the full partition name can be long, a PNR is created in each trail file for the first time the partition is written. Each PNR, contains the partition name and partition object ID.

For primary Extract, PNR is generated only for partition matching and included by `PARTITION` and `PARTITIONEXCLUDE` parameters. DML records from these partitions have an index to the table definition record and another index to the partition name record. DML records from all other tables such as non-partitioned tables or partitioned tables not matching or excluded by the `PARTITION` or `PARTITIONEXCLUDE` parameters, only have an index to the table definition record as done today. For the Distribution Service, the PNR is written if source trail record contains a PNR index.

Viewing the Partition Name and PNR Index in Logdump

Use the Logdump utility to display the partition name record and the DML containing the PNR index.

Here's an example that shows capturing the display in a file:

```
$ logdump > output.txt <<EOF
ghdr on
detail data
open ./dirdat/tr000000000
n 200
EOF
```

The output displays the PNR and the DML with the PNR index values, as shown in the following example:

```
HDR-IND   :      E (X45)          PARTITION   :      . (XFF80)
UNDOFLAG  :      . (X00)          BEFOREAFTER:      A (X41)
RECLNGTH  :      0 (X0000)        IO TIME     : 2019/01/17 16:48:01.129.045
IOTYPE    :     170 (XAA)          ORIGNODE    :      4 (X04)
TRANSIND  :      . (X03)          FORMATTYPE  :      R (X52)
SYSKEYLEN :      0 (X00)          INCOMPLETE :      . (X00)
TDR/PNR IDX: (001, 002)          AUDITPOS    : 13287580
CONTINUED :      N (X00)          RECCOUNT    :      1 (X01)
```

```
2019/01/17 16:48:01.129.045 METADATA          LEN 0 RBA 3425
PARTITION NAME: P1 PARTITION ID: 75,234  FLAGS: X00000001
```

```
HDR-IND   :      E (X45)          PARTITION   :      . (XFF8C)
UNDOFLAG  :      . (X00)          BEFOREAFTER:      A (X41)
RECLNGTH  :     18 (X0012)        IO TIME     : 2019/01/17 16:47:58.000.000
IOTYPE    :      5 (X05)          ORIGNODE    :     255 (XFF)
TRANSIND  :      . (X00)          FORMATTYPE  :      R (X52)
SYSKEYLEN :      0 (X00)          INCOMPLETE :      . (X00)
AUDITRBA  :      15              AUDITPOS    : 13287580
CONTINUED :      N (X00)          RECCOUNT    :      1 (X01)
```

```
2019/01/17 16:47:58.000.000 INSERT          LEN    18 RBA 3486
NAME: TKGGU1.T1 (PARTITION: P1, TDR/PNR INDEX: 1/2)
AFTER IMAGE:                                PARTITION X8C  G  B
0000 0500 0000 0100 3101 0005 0000 0001 0031 | .....1.....1
```

```

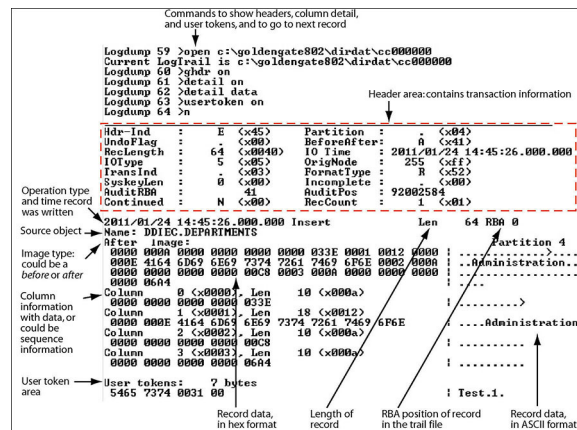
COLUMN      0 (X0000), LEN      5 (X0005)
  0000 0100 31                               | ....1
COLUMN      1 (X0001), LEN      5 (X0005)
  0000 0100 31                               | ....1

```

Example of an Oracle GoldenGate Record

The following illustrates an Oracle GoldenGate record as viewed with Logdump. The first portion (the list of fields) is the header and the second portion is the data area. The record looks similar to this on all platforms supported by Oracle GoldenGate.

Figure 13-1 Example of an Oracle GoldenGate Record



Record Header Area

The Oracle GoldenGate record header provides metadata of the data that is contained in the record and includes the following information.

- The operation type, such as an insert, update, or delete
- The before or after indicator for updates
- Transaction information, such as the transaction group and commit timestamp

Description of Header Fields

The following describes the fields of the Oracle GoldenGate record header. Some fields apply only to certain platforms.

Table: Oracle GoldenGate record header fields

Field	Description
Hdr-Ind	Should always be a value of E, indicating that the record was created by the Extract process. Any other value indicates invalid data.
RecLength	The length, in bytes, of the record buffer.

Field	Description
IOType	The type of operation represented by the record. See Table G-2 - Oracle GoldenGate Operation Types for a list of operation types.
TransInD	The place of the record within the current transaction. Values are: 0 — first record in transaction 1 — neither first nor last record in transaction 2 — last record in the transaction 3 — only record in the transaction
AuditRBA	Identifies the transaction log identifier, such as the Oracle redo log sequence number.
Continued	(Windows and UNIX) Identifies whether or not the record is a segment of a larger piece of data that is too large to fit within one record. LOBs, CLOBs, and some VARCHARs are stored in segments. Unified records that contain both before and after images in a single record (due to the <code>UPDATERECORDFORMAT</code> parameter) may exceed the maximum length of a record and may also generate segments. Y — the record is a segment; indicates to Oracle GoldenGate that this data continues to another record. N — there is no continuation of data to another segment; could be the last in a series or a record that is not a segment of larger data.
Partition	For Windows and UNIX records, this field will always be a value of 4 (FieldComp compressed record in internal format). For these platforms, the term Partition does not indicate that the data represents any particular logical or physical partition within the database structure.
BeforeAfter	Identifies whether the record is a before (B) or after (A) image of an update operation. Records that combine both before and after images as the result of the <code>UPDATERECORDFORMAT</code> parameter are marked as after images. Inserts are always after images, deletes are always before images.
IO Time	The time when the operation occurred, in local time of the source system, in GMT format. This time may be the same or different for every operation in a transaction depending on when the operation occurred.
FormatType	Identifies whether the data was read from the transaction log or fetched from the database. F — fetched from database R — readable in transaction log
Incomplete	This field is obsolete.

Field	Description
AuditPos	Identifies the position in the transaction log of the data.
RecCount	(Windows and UNIX) Used for LOB data when it must be split into chunks to be written to the Oracle GoldenGate file. RecCount is used to reassemble the chunks.

Using Header Data

Some of the data available in the Oracle GoldenGate record header can be used for mapping by using the GGHEADER option of the @GETENV function or by using any of the following transaction elements as the source expression in a COLMAP statement in the TABLE or MAP parameter.

- GGS_TRANS_TIMESTAMP
- GGS_TRANS_RBA
- GGS_OP_TYPE
- GGS_BEFORE_AFTER_IND

Record Data Area

The data area of the Oracle GoldenGate trail record contains the following:

- The time that the change was written to the Oracle GoldenGate file
- The type of database operation
- The length of the record
- The relative byte address within the trail file
- The table name
- The data changes in hex format

The following explains the differences in record image formats used by Oracle GoldenGate on Windows, UNIX, Linux, and NonStop systems.

Full Record Image Format (NonStop Sources)

A full record image contains the values of all of the columns of a processed row. Full record image format is generated in the trail when the source system is HP NonStop, and only when the IOType specified in the record header is one of the following:

3 – Delete 5 – Insert 10 – Update

Each full record image has the same format as if retrieved from a program reading the original file or table directly. For SQL tables, datetime fields, nulls, and other data is written exactly as a program would select it into an application buffer. Although datetime fields are represented

internally as an eight-byte timestamp, their external form can be up to 26 bytes expressed as a string. Enscribe records are retrieved as they exist in the original file.

When the operation type is `Insert` or `Update`, the image contains the contents of the record after the operation (the after image). When the operation type is `Delete`, the image contains the contents of the record before the operation (the before image).

For records generated from an Enscribe database, full record images are output unless the original file has the `AUDITCOMPRESS` attribute set to `ON`. When `AUDITCOMPRESS` is `ON`, compressed update records are generated whenever the original file receives an update operation. (A full image can be retrieved by the Extract process by using the `FETCHCOMPS` parameter.)

Compressed Record Image Format (Windows, UNIX, Linux Sources)

A compressed record image contains only the key (primary, unique, `KEYCOLS`) and the columns that changed in the processed row. By default, trail records written by processes on Windows and UNIX systems are always compressed.

The format of a compressed record is as follows:

```
column_index
      column_length
      column_data[...]
```

Where:

- `column_index`

is the ordinal index of the column within the source table (2 bytes).

- `column_length`

is the length of the data (2 bytes).

- `column_data`

is the data, including

`NULL`

or

`VARCHAR`

length indicators.

Enscribe records written from the NonStop platform may be compressed. The format of a compressed Enscribe record is as follows:

```
field_offset
      field_length field_value[...]
```

Where:

- `field_offset`

is the offset within the original record of the changed value (2 bytes).

- `field_length`

is the length of the data (2 bytes).

- `field_value`

is the data, including

NULL

or

VARCHAR

length indicators.

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

Tokens Area

The trail record also can contain two areas for tokens. One is for internal use and is not documented here, and the other is the user tokens area. User tokens are environment values that are captured and stored in the trail record for replication to target columns or other purposes. If used, these tokens follow the data portion of the record and appear similar to the following when viewed with Logdump:

Parameter	Value
TKN-HOST TKN-GROUP TKN-BA_IND TKN-COMMIT_TS TKN-POS TKN-RBA TKN-TABLE	: syshq : EXTORA : AFTER : 2011-01-24 17:08:59.000000 : 3604496 : 4058 :
TKN-OPTYPE TKN-LENGTH TKN-TRAN_IND	SOURCE.CUSTOMER : INSERT : 57 : BEGIN

Oracle GoldenGate Operation Types

The following are some of the Oracle GoldenGate operation types. Types may be added as new functionality is added to Oracle GoldenGate. For a more updated list, use the `SHOW RECTYPE` command in the Logdump utility:

Type	Description	Platform
1-Abort	A transaction aborted.	NSK TMF
2-Commit	A transaction committed.	NSK TMF

Type	Description	Platform
3-Delete	A record/row was deleted. A <code>Delete</code> record usually contains a full record image. However, if the <code>COMPRESSDELETES</code> parameter was used, then only key columns will be present.	All
4-EndRollback	A database rollback ended	NSK TMF
5-Insert	A record/row was inserted. An <code>Insert</code> record contains a full record image.	All
6-Prepared	A networked transaction has been prepared to commit.	NSK TMF
7-TMF-Shutdown	A TMF shutdown occurred.	NSK TMF
8-TransBegin	No longer used.	NSK TMF
9-TransRelease	No longer used.	NSK TMF
10-Update	A record/row was updated. An <code>Update</code> record contains a full record image. Note: If the partition indicator in the record header is 4, then the record is in <code>FieldComp</code> format (see below) and the update is compressed.	All
11-UpdateComp	A record/row in <code>TMF AuditComp</code> format was updated. In this format, only the changed bytes are present. A 4-byte descriptor in the format of 2-byte_offset2-byte_length precedes each data fragment. The byte offset is the ordinal index of the column within the source table. The length is the length of the data.	NSK TMF
12-FileAlter	An attribute of a database file was altered.	NSK
13-FileCreate	A database file was created.	NSK
14-FilePurge	A database file was deleted.	NSK
15-FieldComp	A row in a SQL table was updated. In this format, only the changed bytes are present. Before images of unchanged columns are not logged by the database. A 4-byte descriptor in the format of 2-byte_offset2-byte_length precedes each data fragment. The byte offset is the ordinal index of the column within the source table. The length is the length of the data. A partition indicator of 4 in the record header indicates <code>FieldComp</code> format.	All
16-FileRename	A file was renamed.	NSK

Type	Description	Platform
17-AuxPointer	Contains information about which AUX trails have new data and the location at which to read.	NSK TMF
18-NetworkCommit	A networked transaction committed.	NSK TMF
19-NetworkAbort	A networked transaction was aborted.	NSK TMF
90-(GGS)SQLCol	A column or columns in a SQL table were added, or an attribute changed.	NSK
100-(GGS)Purgedata	All data was removed from the file (PURGEDATA).	NSK
101-(GGS)Purge(File)	A file was purged.	NSK non-TMF
102-(GGS)Create(File)	A file was created. The Oracle GoldenGate record contains the file attributes.	NSK non-TMF
103-(GGS)Alter(File)	A file was altered. The Oracle GoldenGate record contains the altered file attributes.	NSK non-TMF
104-(GGS)Rename(File)	A file was renamed. The Oracle GoldenGate record contains the original and new names.	NSK non-TMF
105-(GGS)Setmode	A SETMODE operation was performed. The Oracle GoldenGate record contains the SETMODE information.	NSK non-TMF
106-GGSChangeLabel	A CHANGELABEL operation was performed. The Oracle GoldenGate record contains the CHANGELABEL information.	NSK non-TMF
107-(GGS)Control	A CONTROL operation was performed. The Oracle GoldenGate record contains the CONTROL information.	NSK non-TMF
115 and 117 (GGS)KeyFieldComp(32)	A primary key was updated. The Oracle GoldenGate record contains the before image of the key and the after image of the key and the row. The data is in FieldComp format (compressed), meaning that before images of unchanged columns are not logged by the database.	Windows and UNIX
116-LargeObject 116-LOB	Identifies a RAW, BLOB, CLOB, or LOB column. Data of this type is stored across multiple records.	Windows and UNIX
132-(GGS) SequenceOp	Identifies an operation on a sequence.	Windows and UNIX

Type	Description	Platform
134-UNIFIED UPDATE 135-UNIFIED PKUPDATE	Identifies a unified trail record that contains both before and after values in the same record. The before image in a <code>UNIFIED UPDATE</code> contains all of the columns that are available in the transaction record for both the before and after images. The before image in a <code>UNIFIED UPDATE</code> contains 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 <code>UPDATE</code> .	Windows and UNIX
160 - DDL_Op	Identifies a DDL operation	Windows and UNIX
161-RecordFragment	Identifies part of a large row that must be stored across multiple records (more than just the base record).	Windows and UNIX
200-GGSUnstructured Block 200-BulkIO	A <code>BULKIO</code> operation was performed. The Oracle GoldenGate record contains the <code>RAW DP2</code> block.	NSK non-TMF

Type	Description	Platform
201 through 204	<p>These are different types of NonStop trace records. Trace records are used by Oracle GoldenGate support analysts. The following are descriptions.</p> <ul style="list-style-type: none"> ARTYPE_FILECLOSE_GGS 201 — the source application closed a file that was open for unstructured I/O. Used by Replicat ARTYPE_LOGGERTS_GGS 202 — Logger heartbeat record ARTYPE_EXTRACTERTS_GGS S 203 — unused ARTYPE_COLLECTORTS_GGS S 204 — unused 	NSK non-TMF
205-GGSComent	<p>Indicates a comment record created by the Logdump utility. Comment records are created by Logdump at the beginning and end of data that is saved to a file with Logdump's <code>SAVE</code> command.</p>	All

Type	Description	Platform
249 through 254	<p>These are different types of NonStop trace records. Trace records are used by Oracle GoldenGate support analysts. The following are descriptions.</p> <ul style="list-style-type: none"> <p>ARTYPE_LOGGER_ADDED_STATS 249</p> <p>— a stats record created by Logger when the source application closes its open on Logger (if SENDERSTATS is enabled and stats are written to the logtrail)</p> <p>ARTYPE_LIBRARY_OPEN 250</p> <p>— written by BASELIB to show that the application opened a file</p> <p>ARTYPE_LIBRARY_CLOSE 251</p> <p>— written by BASELIB to show that the application closed a file.</p> <p>ARTYPE_LOGGER_ADDED_OPEN 252</p> <p>— unused</p> <p>ARTYPE_LOGGER_ADDED_CLOSE 253 — unused</p> 	NSK non-TM

Type	Description	Platform
	<ul style="list-style-type: none"> ARTYPE_LOGGER_ADDED_INFO 	
	254	
	— written by Logger and contains information about the source application that performed the I/O in the subsequent record (if	
	SENDERSTATS	
	is enabled and stats are written to the logtrail). The file name in the trace record is the object file of the application. The trace data has the application process name and the name of the library (if any) that it was running with.	

Checkpoint Tables Additional Details

When database checkpoints are being used, Oracle GoldenGate creates a checkpoint table with a user-defined name in the database upon execution of the `ADD CHECKPOINTTABLE` command, or a user can create the table by using the `chkpt_db_create.sql` script (where `db` is an abbreviation of the type of database that the script supports).

There are two tables: the main checkpoint table and an auxiliary checkpoint table that is created automatically. The auxiliary table, known as the transaction table, bears the name of the primary checkpoint table appended with `_lox`. Each Replicat, or each thread of a coordinated Replicat, uses one row in the checkpoint table to store its progress information. At checkpoint time, there typically are some number of transactions (among the total `n` transactions) that were applied, and the rest are still in process. For example, if Replicat is processing a group of `n` transactions ranging from `CSN1` to `CSN3`. `CSN1` is the high watermark and `CSN3` is the low watermark. Any transaction with a CSN higher than the high watermark has not been processed, and any transaction with a CSN lower than the low watermark has already been processed. Completed transactions are stored in the `LOG_CMPLT_XID` column of the checkpoint table. Any overflow of these transactions is stored in the transaction table (auxiliary checkpoint table) in the `LOG_CMPLT_XID` column of that table.

Currently, Replicat (or each Replicat thread of a coordinated Replicat) applies transactions serially (not in parallel); therefore, the high watermark (the `LOG_CSN` value in the table) is always the same as the low watermark (the `LOG_CMPLT_CSN` value in the table), and there typically is only one transaction ID in the `LOG_CMPLT_XID` column. The only exception is when there are multiple transactions sharing the same CSN.

Do not change the names or attributes of the columns in these tables. You can change table storage attributes as needed.

Column	Description
LOG_BSN	The LOG_BSN provides information needed to set Extract back in time to reprocess transactions. Some filtering by Replicat is necessary because Extract will likely re-generate a small amount of data that was already applied by Replicat.
VERSION	The version of the checkpoint table format. Enables future enhancements to be identified as version numbers of the table.
AUDIT_TS	The timestamp of the commit of the source transaction.
SEQNO	The sequence number of the input trail that Replicat was reading at the time of the checkpoint.
RBA	The relative byte address that Replicat reached in the trail identified by SEQNO. RBA + SEQNO provide an absolute position in the trail that identifies the progress of Replicat at the time of checkpoint.
GROUP_NAME (primary key)	The name of a Replicat group using this table for checkpoints. There can be multiple Replicat groups using the same table. This column is part of the primary key.
LAST_UPDATE_TS	The date and time when the checkpoint table was last updated.
CREATE_TS	The date and time when the checkpoint table was created.
CURRENT_DIR	The current Oracle GoldenGate home directory or folder.
LOG_CMPLT_XIDS	Stores the transactions between the high and low watermarks that are already applied.
LOG_CMPLT_CSN	Stores the low watermark, or the lower boundary, of the CSNs. Any transaction with a lower CSN than this value has already been processed.
LOG_CSN	Stores the high watermark, or the upper boundary, of the CSNs. Any transaction with a CSN higher than this value has not been processed.
LOG_XID	Not used. Retained for backward compatibility.

Column	Description
GROUP_KEY (primary key)	A unique identifier that, together with GROUPNAME , uniquely identifies a checkpoint regardless of how many Replicat groups are writing to the same table. This column is part of the primary key.
Column	Description
GROUP_KEY	A unique identifier that, together with GROUPNAME, uniquely identifies a checkpoint regardless of how many Replicat groups are writing to the same table. This column is part of the primary key of the transaction table.
LOG_CMPLT_XIDS_SEQ	Creates unique rows in the event there are so many overflow transactions that multiple rows are required to store them all. This column is part of the primary key of the transaction table.
LOG_CMPLT_XIDS	Stores the overflow of transactions between the high and low watermarks that are already applied.
LOG_CMPLT_CSN	The foreign key that references the checkpoint table. This column is part of the primary key of the transaction table.
GROUP_NAME	The name of a Replicat group using this table for checkpoints. There can be multiple Replicat groups using the same table. This column is part of the primary key of the transaction table.

Internal Checkpoint Information

The `INFO` command with the `SHOWCH` option not only displays current checkpoint entries, but it also displays metadata information about the record itself. This information is not documented and is for use by the Oracle GoldenGate processes and by support personnel when resolving a support case.

The metadata is contained in the following entries in the `SHOWCH` output.

Header:

```
Version = 2

Record Source = A

Type = 1

# Input Checkpoints = 1

# Output Checkpoints = 0

File Information:
```

```

Block Size = 2048

Max Blocks = 100

Record Length = 2048

Current Offset = 0

Configuration:

Data Source = 0

Transaction Integrity = -1

Task Type = 0

Status:

Start Time = 2011-01-12 13:10:13

Last Update Time = 2011-01-12 21:23:31

Stop Status = A

Last Result = 400

```

INFO EXTRACT SHOWCH Command: Checkpoint Information

The following sample presents the checkpoint information returned by the `INFO EXTRACT` command with the `SHOWCH` option. In this case, the data source is an Oracle RAC database cluster, so there is thread information included in the output. You can view past checkpoints by specifying the number of them that you want to view after the `SHOWCH` argument.

```

EXTRACT JC108XT Last Started 2011-01-01 14:15 Status ABENDED
Checkpoint Lag 00:00:00 (updated 00:00:01 ago)
Log Read Checkpoint File /orarc/oradata/racq/redo01.log
  2011-01-01 14:16:45 Thread 1, Segno 47, RBA 68748800
Log Read Checkpoint File /orarc/oradata/racq/redo04.log
  2011-01-01 14:16:19 Thread 2, Segno 24, RBA 65657408
Current Checkpoint Detail:
Read Checkpoint #1
Oracle RAC Redo Log
Startup Checkpoint (starting position in data source):
Thread #: 1
Sequence #: 47
RBA: 68548112
Timestamp: 2011-01-01 13:37:51.000000
SCN: 0.8439720
Redo File: /orarc/oradata/racq/redo01.log

Recovery Checkpoint (position of oldest unprocessed transaction in data
source):
Thread #: 1

```

```
Sequence #: 47
RBA: 68748304
Timestamp: 2011-01-01 14:16:45.000000
SCN: 0.8440969
Redo File: /orarc/oradata/racq/redo01.log
Current Checkpoint (position of last record read in the data source):
Thread #: 1
Sequence #: 47
RBA: 68748800
Timestamp: 2011-01-01 14:16:45.000000
SCN: 0.8440969
Redo File: /orarc/oradata/racq/redo01.log
Read Checkpoint #2
Oracle RAC Redo Log
Startup Checkpoint (starting position in data source):
Sequence #: 24
RBA: 60607504
Timestamp: 2011-01-01 13:37:50.000000
SCN: 0.8439719
Redo File: /orarc/oradata/racq/redo04.log
Recovery Checkpoint (position of oldest unprocessed transaction in data
source):
Thread #: 2
Sequence #: 24
RBA: 65657408
Timestamp: 2011-01-01 14:16:19.000000
SCN: 0.8440613
Redo File: /orarc/oradata/racq/redo04.log
Current Checkpoint (position of last record read in the data source):
Thread #: 2
Sequence #: 24
RBA: 65657408
Timestamp: 2011-01-01 14:16:19.000000
SCN: 0.8440613
Redo File: /orarc/oradata/racq/redo04.log
Write Checkpoint #1
GGT Log Trail
Current Checkpoint (current write position):
Sequence #: 2
RBA: 2142224
Timestamp: 2011-01-01 14:16:50.567638
Extract Trail: ./dirdat/eh
Header:
Version = 2
Record Source = A
Type = 6
# Input Checkpoints = 2
# Output Checkpoints = 1
File Information:
Block Size = 2048
Max Blocks = 100
Record Length = 2048
Current Offset = 0
Configuration:
Data Source = 3
Transaction Integrity = 1
```

```
Task Type = 0
Status:
Start Time = 2011-01-01 14:15:14
Last Update Time = 2011-01-01 14:16:50
Stop Status = A
Last Result = 400
```

INFO REPLICAT, SHOWCH: Checkpoint Information

The basic command shows current checkpoints. To view a specific number of previous checkpoints, type the value after the `SHOWCH` argument.

```
REPLICAT JC108RP Last Started 2011-01-12 13:10 Status RUNNING
Checkpoint Lag 00:00:00 (updated 111:46:54 ago)
Log Read Checkpoint File ./dirdat/eh000000000
First Record RBA 3702915
Current Checkpoint Detail:
Read Checkpoint #1
GGS Log Trail
Startup Checkpoint(starting position in data source):
Sequence #: 0
RBA: 3702915
Timestamp: Not Available
Extract Trail: ./dirdat/eh
Current Checkpoint (position of last record read in the data source):
Sequence #: 0
RBA: 3702915
Timestamp: Not Available
Extract Trail: ./dirdat/eh
Header:
Version = 2
Record Source = A
Type = 1
# Input Checkpoints = 1
# Output Checkpoints =
File Information:
Block Size = 2048
Max Blocks = 100
Record Length = 2048
Current Offset = 0
Configuration:
Data Source = 0
Transaction Integrity = -1
Task Type = 0
Status:
Start Time = 2011-01-12 13:10:13
Last Update Time = 2011-01-12 21:23:31
Stop Status = A
Last Result = 400
```

Supported Character Sets

Here's a list of character sets that Oracle GoldenGate supports when converting data from source to target.

The identifiers that are shown should be used for Oracle GoldenGate parameters or commands when a character set must be specified, instead of the actual character set name. Currently Oracle GoldenGate does not provide a facility to specify the database-specific character set.

Supported Character Sets - Oracle

Table 13-1 Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
al32utf8	Unicode 9.0 Universal Character Set (UCS), UTF-8 encoding scheme
ar8ados710t	Arabic MS-DOS 710 8-bit Latin/Arabic
ar8ados710	Arabic MS-DOS 710 Server 8-bit Latin/Arabic
ar8ados720t	Arabic MS-DOS 720 8-bit Latin/Arabic
ar8ados720	Arabic MS-DOS 720 Server 8-bit Latin/Arabic
ar8aptec715t	APTEC 715 8-bit Latin/Arabic
ar8aptec715	APTEC 715 Server 8-bit Latin/Arabic
ar8arabicmacs	Mac Server 8-bit Latin/Arabic
ar8arabicmact	Mac 8-bit Latin/Arabic
ar8arabicmac	Mac Client 8-bit Latin/Arabic
ar8asmo708plus	ASMO 708 Plus 8-bit Latin/Arabic
ar8asmo8x	ASMO Extended 708 8-bit Latin/Arabic
ar8ebcdic420s	EBCDIC Code Page 420 Server 8-bit Latin/Arabic
ar8ebcdicx	EBCDIC XBASIC Server 8-bit Latin/Arabic
ar8hparabic8t	HP 8-bit Latin/Arabic
ar8iso8859p6	ISO 8859-6 Latin/Arabic
ar8mswin1256	MS Windows Code Page 1256 8-Bit Latin/Arabic
ar8mussad768t	Mussa'd Alarabi/2 768 8-bit Latin/Arabic
ar8mussad768	Mussa'd Alarabi/2 768 Server 8-bit Latin/Arabic
ar8nafitha711t	Nafitha International 711 Server 8-bit Latin/Arabic
ar8nafitha711	Nafitha Enhanced 711 Server 8-bit Latin/Arabic
ar8nafitha721t	Nafitha International 721 8-bit Latin/Arabic
ar8nafitha721	Nafitha International 721 Server 8-bit Latin/Arabic
ar8sakhr706	SAKHR 706 Server 8-bit Latin/Arabic
ar8sakhr707t	SAKHR 707 8-bit Latin/Arabic
ar8sakhr707	SAKHR 707 Server 8-bit Latin/Arabic
ar8xbasic	XBASIC 8-bit Latin/Arabic

Table 13-1 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
az8iso8859p9e	ISO 8859-9 Azerbaijani
bg8mswin	MS Windows 8-bit Bulgarian Cyrillic
bg8pc437s	IBM-PC Code Page 437 8-bit (Bulgarian Modification)
blt8cp921	Latvian Standard LVS8-92(1) Windows/Unix 8-bit Baltic
blt8ebcdic1112s	EBCDIC Code Page 1112 8-bit Server Baltic Multilingual
blt8ebcdic1112	EBCDIC Code Page 1112 8-bit Baltic Multilingual
blt8iso8859p13	ISO 8859-13 Baltic
blt8mswin1257	MS Windows Code Page 1257 8-bit Baltic
blt8pc775	IBM-PC Code Page 775 8-bit Baltic
bn8bscii	Bangladesh National Code 8-bit BSCII
cdn8pc863	IBM-PC Code Page 863 8-bit Canadian French
ce8bs2000	Siemens EBCDIC.DF.04-2 8-bit Central European
cel8iso8859p14	ISO 8859-13 Celtic
ch7dec	DEC VT100 7-bit Swiss (German/French)
cl8bs2000	Siemens EBCDIC.EHC.LC 8-bit Latin/Cyrillic-1
cl8ebcdic1025c	EBCDIC Code Page 1025 Client 8-bit Cyrillic
cl8ebcdic1025r	EBCDIC Code Page 1025 Server 8-bit Cyrillic
cl8ebcdic1025s	EBCDIC Code Page 1025 Server 8-bit Cyrillic
cl8ebcdic1025	EBCDIC Code Page 1025 8-bit Cyrillic
cl8ebcdic1025x	EBCDIC Code Page 1025 (Modified) 8-bit Cyrillic
cl8ebcdic1158r	EBCDIC Code Page 1158 Server 8-bit Cyrillic
cl8ebcdic1158	EBCDIC Code Page 1158 8-bit Cyrillic
cl8iso8859p5	ISO 8859-5 Latin/Cyrillic
cl8isoir111	SOIR111 Cyrillic
cl8koi8r	RELCOM Internet Standard 8-bit Latin/Cyrillic
cl8koi8u	KOI8 Ukrainian Cyrillic
cl8maccyrillics	Mac Server 8-bit Latin/Cyrillic
cl8maccyrillic	Mac Client 8-bit Latin/Cyrillic
cl8mswin1251	MS Windows Code Page 1251 8-bit Latin/Cyrillic
d7dec	DEC VT100 7-bit German
d7siemens9780x	Siemens 97801/97808 7-bit German
d8bs2000	Siemens 9750-62 EBCDIC 8-bit German
d8ebcdic1141	EBCDIC Code Page 1141 8-bit Austrian German
d8ebcdic273	EBCDIC Code Page 273/1 8-bit Austrian German

Table 13-1 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
dk7siemens9780x	Siemens 97801/97808 7-bit Danish
dk8bs2000	Siemens 9750-62 EBCDIC 8-bit Danish
dk8ebcdic1142	EBCDIC Code Page 1142 8-bit Danish
dk8ebcdic277	EBCDIC Code Page 277/1 8-bit Danish
e7dec	DEC VT100 7-bit Spanish
e7siemens9780x	Siemens 97801/97808 7-bit Spanish
e8bs2000	Siemens 9750-62 EBCDIC 8-bit Spanish
ee8bs2000	Siemens EBCDIC.EHC.L2 8-bit East European
ee8ebcdic870c	EBCDIC Code Page 870 Client 8-bit East European
ee8ebcdic870s	EBCDIC Code Page 870 Server 8-bit East European
ee8ebcdic870	EBCDIC Code Page 870 8-bit East European
ee8iso8859p2	ISO 8859-2 East European
ee8maccess	Mac Server 8-bit Central European
ee8macce	Mac Client 8-bit Central European
ee8maccroatians	Mac Server 8-bit Croatian
ee8maccroatian	Mac Client 8-bit Croatian
ee8mswin1250	MS Windows Code Page 1250 8-bit East European
ee8pc852	IBM-PC Code Page 852 8-bit East European
eec8euroasci	EEC Targon 35 ASCII West European/Greek
eec8europa3	EEC EUROPA3 8-bit West European/Greek
el8dec	DEC 8-bit Latin/Greek
el8ebcdic423r	IBM EBCDIC Code Page 423 for RDBMS server-side
el8ebcdic875r	EBCDIC Code Page 875 Server 8-bit Greek
el8ebcdic875s	EBCDIC Code Page 875 Server 8-bit Greek
el8ebcdic875	EBCDIC Code Page 875 8-bit Greek
el8gcos7	Bull EBCDIC GCOS7 8-bit Greek
el8iso8859p7	ISO 8859-7 Latin/Greek
el8macgreeks	Mac Server 8-bit Greek
el8macgreek	Mac Client 8-bit Greek
el8mswin1253	MS Windows Code Page 1253 8-bit Latin/Greek
el8pc437s	IBM-PC Code Page 437 8-bit (Greek modification)
el8pc737	IBM-PC Code Page 737 8-bit Greek/Latin
el8pc851	IBM-PC Code Page 851 8-bit Greek/Latin
el8pc869	IBM-PC Code Page 869 8-bit Greek/Latin

Table 13-1 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
et8mswin923	MS Windows Code Page 923 8-bit Estonian
f7dec	DEC VT100 7-bit French
f7siemens9780x	Siemens 97801/97808 7-bit French
f8bs2000	Siemens 9750-62 EBCDIC 8-bit French
f8ebcdic1147	EBCDIC Code Page 1147 8-bit French
f8ebcdic297	EBCDIC Code Page 297 8-bit French
hu8abmod	Hungarian 8-bit Special AB Mod
hu8cwi2	Hungarian 8-bit CWI-2
i7dec	DEC VT100 7-bit Italian
i7siemens9780x	Siemens 97801/97808 7-bit Italian
i8ebcdic1144	EBCDIC Code Page 1144 8-bit Italian
i8ebcdic280	EBCDIC Code Page 280/1 8-bit Italian
in8iscii	Multiple-Script Indian Standard 8-bit Latin/Indian
is8macicelandics	Mac Server 8-bit Icelandic
is8macicelandic	Mac Client 8-bit Icelandic
is8pc861	IBM-PC Code Page 861 8-bit Icelandic
iw7is960	Israeli Standard 960 7-bit Latin/Hebrew
iw8ebcdic1086	EBCDIC Code Page 1086 8-bit Hebrew
iw8ebcdic424s	EBCDIC Code Page 424 Server 8-bit Latin/Hebrew
iw8ebcdic424	EBCDIC Code Page 424 8-bit Latin/Hebrew
iw8iso8859p8	ISO 8859-8 Latin/Hebrew
iw8machebrews	Mac Server 8-bit Hebrew
iw8machebrew	Mac Client 8-bit Hebrew
iw8mswin1255	MS Windows Code Page 1255 8-bit Latin/Hebrew
iw8pc1507	IBM-PC Code Page 1507/862 8-bit Latin/Hebrew
ja16dbcs	IBM EBCDIC 16-bit Japanese
ja16ebcdic930	IBM DBCS Code Page 290 16-bit Japanese
ja16euctilde	Same as ja16euc except for the way that the wave dash and the tilde are mapped to and from Unicode
ja16euc	EUC 24-bit Japanese
ja16eucyen	EUC 24-bit Japanese with '¥' mapped to the Japanese yen character
ja16macsjis	Mac client Shift-JIS 16-bit Japanese
ja16sjistilde	Same as ja16sjis except for the way that the wave dash and the tilde are mapped to and from Unicode.
ja16sjis	Shift-JIS 16-bit Japanese

Table 13-1 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
ja16sjisyen	Shift-JIS 16-bit Japanese with '\ ' mapped to the Japanese yen character
ja16vms	JVMS 16-bit Japanese
ko16dbcs	IBM EBCDIC 16-bit Korean
ko16ksc5601	KSC5601 16-bit Korean
ko16ksccs	KSCCS 16-bit Korean
ko16mswin949	MS Windows Code Page 949 Korean
la8iso6937	ISO 6937 8-bit Coded Character Set for Text Communication
la8passport	German Government Printer 8-bit All-European Latin
lt8mswin921	MS Windows Code Page 921 8-bit Lithuanian
lt8pc772	IBM-PC Code Page 772 8-bit Lithuanian (Latin/Cyrillic)
lt8pc774	IBM-PC Code Page 774 8-bit Lithuanian (Latin)
lv8pc1117	IBM-PC Code Page 1117 8-bit Latvian
lv8pc81r	Latvian Version IBM-PC Code Page 866 8-bit Latin/Cyrillic
lv8rst104090	IBM-PC Alternative Code Page 8-bit Latvian (Latin/Cyrillic)
n7siemens9780x	Siemens 97801/97808 7-bit Norwegian
n8pc865	IBM-PC Code Page 865 8-bit Norwegian
ndk7dec	DEC VT100 7-bit Norwegian/Danish
ne8iso8859p10	ISO 8859-10 North European
nee8iso8859p4	ISO 8859-4 North and North-East European
nl7dec	DEC VT100 7-bit Dutch
ru8besta	BESTA 8-bit Latin/Cyrillic
ru8pc855	IBM-PC Code Page 855 8-bit Latin/Cyrillic
ru8pc866	IBM-PC Code Page 866 8-bit Latin/Cyrillic
s7dec	DEC VT100 7-bit Swedish
s7siemens9780x	Siemens 97801/97808 7-bit Swedish
s8bs2000	Siemens 9750-62 EBCDIC 8-bit Swedish
s8ebcdic1143	EBCDIC Code Page 1143 8-bit Swedish
s8ebcdic278	EBCDIC Code Page 278/1 8-bit Swedish
se8iso8859p3	ISO 8859-3 South European
sf7ascii	ASCII 7-bit Finnish
sf7dec	DEC VT100 7-bit Finnish
th8macthais	Mac Server 8-bit Latin/Thai
th8macthai	Mac Client 8-bit Latin/Thai
th8tisascii	Thai Industrial Standard 620-2533 - ASCII 8-bit

Table 13-1 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
th8tisebcdics	Thai Industrial Standard 620-2533 - EBCDIC Server 8-bit
th8tisebcdic	Thai Industrial Standard 620-2533 - EBCDIC 8-bit
tr7dec	DEC VT100 7-bit Turkish
tr8dec	DEC 8-bit Turkish
tr8ebcdic1026s	EBCDIC Code Page 1026 Server 8-bit Turkish
tr8ebcdic1026	EBCDIC Code Page 1026 8-bit Turkish
tr8macturkishs	Mac Server 8-bit Turkish
tr8macturkish	Mac Client 8-bit Turkish
tr8mswin1254	MS Windows Code Page 1254 8-bit Turkish
tr8pc857	IBM-PC Code Page 857 8-bit Turkish
us7ascii	ASCII 7-bit American
us8bs2000	Siemens 9750-62 EBCDIC 8-bit American
us8icl	ICL EBCDIC 8-bit American
us8pc437	IBM-PC Code Page 437 8-bit American
vn8mswin1258	MS Windows Code Page 1258 8-bit Vietnamese
vn8vn3	VN3 8-bit Vietnamese
we8bs2000e	Siemens EBCDIC.DF.04-F 8-bit West European with Euro symbol
we8bs200015	Siemens EBCDIC.DF.04-9 8-bit WE & Turkish
we8bs2000	Siemens EBCDIC.DF.04-1 8-bit West European
we8dec	DEC 8-bit West European
we8dg	DG 8-bit West European
we8ebcdic1047e	Latin 1/Open Systems 1047
we8ebcdic1047	EBCDIC Code Page 1047 8-bit West European
we8ebcdic1140c	EBCDIC Code Page 1140 Client 8-bit West European
we8ebcdic1140	EBCDIC Code Page 1140 8-bit West European
we8ebcdic1145	EBCDIC Code Page 1145 8-bit West European
we8ebcdic1146	EBCDIC Code Page 1146 8-bit West European
we8ebcdic1148c	EBCDIC Code Page 1148 Client 8-bit West European
we8ebcdic1148	EBCDIC Code Page 1148 8-bit West European
we8ebcdic284	EBCDIC Code Page 284 8-bit Latin American/Spanish
we8ebcdic285	EBCDIC Code Page 285 8-bit West European
we8ebcdic37c	EBCDIC Code Page 37 8-bit Oracle/c
we8ebcdic37	EBCDIC Code Page 37 8-bit West European
we8ebcdic500c	EBCDIC Code Page 500 8-bit Oracle/c

Table 13-1 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
we8ebcdic500	EBCDIC Code Page 500 8-bit West European
we8ebcdic871	EBCDIC Code Page 871 8-bit Icelandic
we8ebcdic924	Latin 9 EBCDIC 924
we8gcos7	Bull EBCDIC GCOS7 8-bit West European
we8hp	HP LaserJet 8-bit West European
we8icl	ICL EBCDIC 8-bit West European
we8iso8859p15	ISO 8859-15 West European
we8iso8859p1	ISO 8859-1 West European
we8iso8859p9	ISO 8859-9 West European & Turkish
we8isoicluk	ICL special version ISO8859-1
we8macroman8s	Mac Server 8-bit Extended Roman8 West European
we8macroman8	Mac Client 8-bit Extended Roman8 West European
we8mswin1252	MS Windows Code Page 1252 8-bit West European
we8ncr4970	NCR 4970 8-bit West European
we8nextstep	NeXTSTEP PostScript 8-bit West European
we8pc850	IBM-PC Code Page 850 8-bit West European
we8pc858	IBM-PC Code Page 858 8-bit West European
we8pc860	IBM-PC Code Page 860 8-bit West European
we8roman8	HP Roman8 8-bit West European
yug7ascii	ASCII 7-bit Yugoslavian
zhs16cgb231280	CGB2312-80 16-bit Simplified Chinese
zhs16dbcs	IBM EBCDIC 16-bit Simplified Chinese
zhs16gbk	GBK 16-bit Simplified Chinese
zhs16maccgb231280	Mac client CGB2312-80 16-bit Simplified Chinese
zht16big5	BIG5 16-bit Traditional Chinese
zht16ccdc	HP CCDC 16-bit Traditional Chinese
zht16dbcs	IBM EBCDIC 16-bit Traditional Chinese
zht16dbt	Taiwan Taxation 16-bit Traditional Chinese
zht16hkscs31	MS Windows Code Page 950 with Hong Kong Supplementary Character Set HKSCS-2001 (character set conversion to and from Unicode is based on Unicode 3.1)
zht16hkscs	MS Windows Code Page 950 with Hong Kong Supplementary Character Set HKSCS-2001 (character set conversion to and from Unicode is based on Unicode 3.0)
zht16mswin950	MS Windows Code Page 950 Traditional Chinese
zht32euc	EUC 32-bit Traditional Chinese

Table 13-1 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
zht32sops	SOPS 32-bit Traditional Chinese
zht32tris	TRIS 32-bit Traditional Chinese

Supported Character Sets - Non-Oracle

Identifier to use in parameter files and commands	Character set
UTF-8	ISO-10646 UTF-8, surrogate pairs are 4 bytes per character
UTF-16	ISO-10646 UTF-16
UTF-16BE	UTF-16 Big Endian
UTF-16LE	UTF-16 Little Endian
UTF-32	ISO-10646 UTF-32
UTF-32BE	UTF-32 Big Endian
UTF-32LE	UTF-32 Little Endian
CESU-8	Similar to UTF-8, correspond to UCS-2 and surrogate pairs are 6 bytes per character
US-ASCII	US-ASCII, ANSI X34-1986
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

Identifier to use in parameter files and commands	Character set
windows-1257	Windows Baltic
windows-1258	Windows Vietnam
windows-874	Windows Thai
cp437	DOS Latin-1
ibm-720	DOS Arabic
cp737	DOS Greek
cp775	DOS Baltic
cp850	DOS multilingual
cp851	DOS Greek-1
cp852	DOS Latin-2
cp855	DOS Cyrillic
cp856	DOS Cyrillic / IBM
cp857	DOS Turkish
cp858	DOS Multilingual with Euro
cp860	DOS Portuguese
cp861	DOS Icelandic
cp862	DOS Hebrew
cp863	DOS French
cp864	DOS Arabic
cp865	DOS Nordic
cp866	DOS Cyrillic / GOST 19768-87

Identifier to use in parameter files and commands	Character set
ibm-867	DOS Hebrew / IBM
cp868	DOS Urdu
cp869	DOS Greek-2
ISO-8859-1	ISO-8859-1 Latin-1/Western Europe
ISO-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-8859-11	ISO-8859-11 Latin/Thai
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
IBM037	IBM 037-1/697-1 EBCDIC, Brazil, Canada, Netherlands, Portugal, US, and 037/1175 Traditional Chinese
IBM01140	IBM 1140-1/695-1 EBCDIC, Brazil, Canada, Netherlands, Portugal, US, and 1140/1175 Traditional Chinese
IBM273	IBM 273-1/697-1 EBCDIC, Austria, Germany
IBM01141	IBM 1141-1/695-1 EBCDIC, Austria, Germany

Identifier to use in parameter files and commands	Character set
IBM277	IBM 277-1/697-1 EBCDIC, Denmark, Norway
IBM01142	IBM 1142-1/695-1 EBCDIC, Denmark, Norway
IBM278	IBM 278-1/697-1 EBCDIC, Finland, Sweden
IBM01143	IBM 1143-1/695-1 EBCDIC, Finland, Sweden
IBM280	IBM 280-1/697-1 EBCDIC, Italy
IBM01144	IBM 1144-1/695-1 EBCDIC, Italy
IBM284	IBM 284-1/697-1 EBCDIC, Latin America, Spain
IBM01145	IBM 1145-1/695-1 EBCDIC, Latin America, Spain
IBM285	IBM 285-1/697-1 EBCDIC, United Kingdom
IBM01146	IBM 1146-1/695-1 EBCDIC, United Kingdom
IBM290	IBM 290 EBCDIC, Japan (Katakana) Extended
IBM297	IBM 297-1/697-1 EBCDIC, France
IBM01147	IBM 1147-1/695-1 EBCDIC, France
IBM420	IBM 420 EBCDIC, Arabic Bilingual
IBM424	IBM 424/941 EBCDIC, Israel (Hebrew - Bulletin Code)
IBM500	IBM 500-1/697-1 EBCDIC, International
IBM01148	IBM 1148-1/695-1 EBCDIC International
IBM870	IBM 870/959 EBCDIC, Latin-2 Multilingual
IBM871	IBM 871-1/697-1 EBCDIC Iceland
IBM918	IBM EBCDIC code page 918, Arabic 2
IBM1149	IBM 1149-1/695-1, EBCDIC Iceland

Identifier to use in parameter files and commands	Character set
IBM1047	IBM 1047/103 EBCDIC, Latin-1 (Open Systems)
ibm-803	IBM 803 EBCDIC, Israel (Hebrew - Old Code)
IBM875	IBM 875 EBCDIC, Greece
ibm-924	IBM 924-1/1353-1 EBCDIC International
ibm-1153	IBM 1153/1375 EBCDIC, Latin-2 Multilingual
ibm-1122	IBM 1122/1037 EBCDIC, Estonia
ibm-1157	IBM 1157/1391 EBCDIC, Estonia
ibm-1112	IBM 1112/1035 EBCDIC, Latvia, Lithuania
ibm-1156	IBM 1156/1393 EBCDIC, Latvia, Lithuania
ibm-4899	IBM EBCDIC code page 4899, Hebrew with Euro
ibm-12712	IBM 12712 EBCDIC, Hebrew (max set including Euro)
ibm-1097	IBM 1097 EBCDIC, Farsi
ibm-1018	IBM 1018 EBCDIC, Finland Sweden (ISO-7)
ibm-1132	IBM 1132 EBCDIC, Laos
ibm-1137	IBM EBCDIC code page 1137, Devanagari
ibm-1025	IBM 1025/1150 EBCDIC, Cyrillic
ibm-1154	IBM EBCDIC code page 1154, Cyrillic with Euro
IBM1026	IBM 1026/1152 EBCDIC, Latin-5 Turkey
ibm-1155	IBM EBCDIC code page 1155, Turkish with Euro
ibm-1123	IBM 1123 EBCDIC, Ukraine
ibm-1158	IBM EBCDIC code page 1158, Ukrainian with Euro

Identifier to use in parameter files and commands	Character set
IBM838	IBM 838/1173 EBCDIC, Thai
ibm-1160	IBM EBCDIC code page 1160, Thai with Euro
ibm-1130	IBM 1130 EBCDIC, Vietnam
ibm-1164	IBM EBCDIC code page 1164, Vietnamese with Euro
ibm-4517	IBM EBCDIC code page 4517, Arabic French
ibm-4971	IBM EBCDIC code page 4971, Greek
ibm-9067	IBM EBCDIC code page 9067, Greek 2005
ibm-16804	IBM EBCDIC code page 16804, Arabic
KOI8-R	Russian and Cyrillic (KOI8-R)
KOI8-U	Ukranian (KOI8-U)
eucTH	EUC Thai
ibm-1162	Windows Thai with Euro
DEC-MCS	DEC Multilingual
hp-roman8	HP Latin-1 Roman8
ibm-901	IBM Baltic ISO-8 CCSID 901
ibm-902	IBM Estonia ISO-8 with Euro CCSID 902
ibm-916	IBM ISO8859-8 CCSID
ibm-922	IBM Estonia ISO-8 CCSID 922
ibm-1006	IBM Urdu ISO-8 CCSID 1006
ibm-1098	IBM Farsi PC CCSID 1098
ibm-1124	Ukranian ISO-8 CCSID 1124

Identifier to use in parameter files and commands	Character set
ibm-1125	Ukranian without Euro CCSID 1125
ibm-1129	IBM Vietnamese without Euro CCSID 1129
ibm-1131	IBM Belarusi CCSID 1131
ibm-1133	IBM Lao CCSID 1133
ibm-4909	IBM Greek Latin ASCII CCSID 4909
JIS_X201	JIS X201 Japanese
windows-932	Windows Japanese
windows-936	Windows Simplified Chinese
ibm-942	IBM Windows Japanese
windows-949	Windows Korean
windows-950	Windows Traditional Chinese
eucjis	EUC Japanese
EUC-JP	IBM/MS EUC Japanese
EUC-CN	EUC Simplified Chinese, GBK
EUC-KR	EUC Korean
EUC-TW	EUC Traditional Chinese
ibm-930	IBM 930/5026 Japanese
ibm-933	IBM 933 Korean
ibm-935	IBM 935 Simplified Chinese
ibm-937	IBM 937 Traditional Chinese
ibm-939	IBM 939/5035 Japanese

Identifier to use in parameter files and commands	Character set
ibm-1364	IBM 1364 Korean
ibm-1371	IBM 1371 Traditional Chinese
ibm-1388	IBM 1388 Simplified Chinese
ibm-1390	IBM 1390 Japanese
ibm-1399	IBM 1399 Japanese
ibm-5123	IBM CCSID 5123 Japanese
ibm-8482	IBM CCSID 8482 Japanese
ibm-13218	IBM CCSID 13218 Japanese
ibm-16684	IBM CCSID 16684 Japanese
shiftjis	Japanese Shift JIS, Tilde 0x8160 mapped to U+301C
gb18030	GB-18030
GB2312	GB-2312-1980
GBK	GBK
HZ	HZ GB2312
Ibm-1381	IBM CCSID 1381 Simplified Chinese
Big5	Big5, Traditional Chinese
Big5-HKSCS	Big5, HongKong ext.
Big5-HKSCS2001	Big5, HongKong ext. HKSCS-2001
ibm-950	IBM Big5, CCSID 950
ibm-949	CCSID 949 Korean
ibm-949C	IBM CCSID 949 Korean, has backslash

Identifier to use in parameter files and commands	Character set
ibm-971	IBM CCSID 971 Korean EUC, KSC5601 1989
x-IBM1363	IBM CCSID 1363, Korean

Supported Locales

Here's a list of the locales that are supported by Oracle GoldenGate. The locale is used when comparing case-insensitive object names.

```
af
af_NA
af_ZA
am
am_ET
ar
ar_AE
ar_BH
ar_DZ
ar_EG
ar_IQ
ar_JO
ar_KW
ar_LB
ar_LY
ar_MA
ar_OM
ar_QA
ar_SA
ar_SD
ar_SY
ar_TN
ar_YE
as
as_IN
az
az_Cyrl
az_Cyrl_AZ
az_Latn
az_Latn_AZ
be
be_BY
bg
bg_BG
bn
```

bn_BD
bn_IN
ca
ca_ES
cs
cs_CZ
cy
cy_GB
da
da_DK
de
de_AT
de_BE
de_CH
de_DE
de_LI
de_LU
el
el_CY
el_GR
en
en_AU
en_BE
en_BW
en_BZ
en_CA
en_GB
en_HK
en_IE
en_IN
en_JM
en_MH
en_MT
en_NA
en_NZ
en_PH
en_PK
en_SG
en_TT
en_US
en_US_POSIX
en_VI
en_ZA
en_ZW
eo
es
es_AR
es_BO
es_CL
es_CO

es_CR
es_DO
es_EC
es_ES
es_GT
es_HN
es_MX
es_NI
es_PA
es_PE
es_PR
es_PY
es_SV
es_US
es_UY
es_VE
et
et_EE
eu
eu_ES
fa
fa_AF
fa_IR
fi
fi_FI
fo
fo_FO
fr
fr_BE
fr_CA
fr_CH
fr_FR
fr_LU
fr_MC
ga
ga_IE
gl
gl_ES
gu
gu_IN
gv
gv_GB
haw
haw_US
he
he_IL
hi
hi_IN
hr
hr_HR

hu
hu_HU
hy
hy_AM
hy_AM_REVISED
id
id_ID
is
is_IS
it
it_CH
it_IT
ja
ja_JP
ka
ka_GE
kk
kk_KZ
kl
kl_GL
km
km_KH
kn
kn_IN
ko
ko_KR
kok
kok_IN
kw
kw_GB
lt
lt_LT
lv
lv_LV
mk
mk_MK
ml
ml_IN
mr
mr_IN
ms
ms_BN
ms_MY
mt
mt_MT
nb
nb_NO
nl
nl_BE
nl_NL

nn
nn_NO
om
om_ET
om_KE
or
or_IN
pa
pa_Guru
pa_Guru_IN
pl
pl_PL
ps
ps_AF
pt
pt_BR
pt_PT
ro
ro_RO
ru
ru_RU
ru_UA
sk
sk_SK
sl
sl_SI
so
so_DJ
so_ET
so_KE
so_SO
sq
sq_AL
sr
sr_Cyrl
sr_Cyrl_BA
sr_Cyrl_ME
sr_Cyrl_RS
sr_Latn
sr_Latn_BA
sr_Latn_ME
sr_Latn_RS
sv
sv_FI
sv_SE
sw
sw_KE
sw_TZ
ta
ta_IN

```
te
te_IN
th
th_TH
ti
ti_ER
ti_ET
tr
tr_TR
uk
uk_UA
ur
ur_IN
ur_PK
uz
uz_Arab
uz_Arab_AF
uz_Cyrl
uz_Cyrl_UZ
uz_Latn
uz_Latn_UZ
vi
vi_VN
zh
zh_Hans
zh_Hans_CN
zh_Hans_SG
zh_Hant
zh_Hant_HK
zh_Hant_MO
zh_Hant_TW
```

Commit Sequence Number (CSN)

When working with Oracle GoldenGate, you might need to refer to a Commit Sequence Number (CSN). A CSN is an identifier that Oracle GoldenGate constructs to identify a transaction for the purpose of maintaining transactional consistency and data integrity. It uniquely identifies a point in time in which a transaction commits to the database.

The CSN can be required to position Extract in the transaction log, to reposition Replicat in the trail, or for other purposes. It is returned by some conversion functions and is included in reports and certain command output.

A CSN is a monotonically increasing identifier generated by Oracle GoldenGate that uniquely identifies a point in time when a transaction commits to the database. Its purpose is to ensure transactional consistency and data integrity as transactions are replicated from source to target. Each kind of database management system generates some kind of unique serial number of its own at the completion of each transaction, which uniquely identifies the commit of that transaction. For example, the Oracle RDBMS generates a System Change Number, which is a monotonically increasing sequence number assigned to every event by Oracle RDBMS. The CSN captures this same identifying information and represents it internally as a series of bytes, but the CSN is processed in a platform-independent manner. A comparison of

any two CSN numbers, each of which is bound to a transaction-commit record in the same log stream, reliably indicates the order in which the two transactions completed.

The CSN is cross-checked with the transaction ID (displayed as XID in Oracle GoldenGate informational output). The XID-CSN combination uniquely identifies a transaction even in cases where there are multiple transactions that commit at the same time, and thus have the same CSN. For example, this can happen in an Oracle RAC environment, where there is parallelism and high transaction concurrency.

The CSN value is stored as a token in any trail record that identifies the commit of a transaction. This value can be retrieved with the `@GETENV` column conversion function and viewed with the Logdump utility.

Using the Commit Sequence Number

This appendix contains information about using the Oracle GoldenGate Commit Sequence Number (CSN) with Oracle and non-Oracle databases.

All database platforms except Oracle, Db2 LUW, and Db2 z/OS have fixed-length CSNs, which are padded with leading zeroes as required to fill the fixed length. CSNs that contain multiple fields can be padded within each field. For more information on CSN, see in Overview: Commit Sequence Number (CSN) in the *Oracle GoldenGate Microservices* guide.

MySQL does not create a transaction ID as part of its event data, so Oracle GoldenGate considers a unique transaction identifier to be a *combination* of the following:

- the log file number of the log file that contains the `START TRANSACTION` record for the transaction that is being identified
- the record offset of that record

Table 13-2 Oracle GoldenGate CSN Values Per Database

Database	CSN Value
Db2 for i	<i>sequence_number</i> Where: <ul style="list-style-type: none">• <i>sequence_number</i> is the fixed-length, 20 digit, decimal-based Db2 for i journal sequence number. Example: 12345678901234567890
Db2 LUW	LRI Where: For version 10.1 and later, <i>LRI</i> is a period-separated pair of numbers for the Db2 log record identifier. Example: 123455.34645

Table 13-2 (Cont.) Oracle GoldenGate CSN Values Per Database


Database	CSN Value
Db2 z/OS	<p><i>LSN</i></p> <p>where:</p> <ul style="list-style-type: none">LSN is, up to 20 hexadecimal digit representation of the 10 byte LSN in the transaction log. <div> Note: Oracle GoldenGate uses LSN to represent both the non-data sharing LSN and data sharing LRSN as the format is same.</div> <p>Example:</p> <p>0x1A3367F6BA12289</p>
MySQL	<p><i>LogNum:LogPosition</i></p> <p>Where:</p> <ul style="list-style-type: none"><i>LogNum</i> is the the name of the log file that contains the START TRANSACTION record for the transaction that is being identified.<i>LogPosition</i> is the event offset value of that record. Event offset values are stored in the record header section of a log record. <p>For example, if the log number is 12 and the log position is 121, the CSN is:</p> <p>000012:0000000000000121</p>
MySQL (Group Replication)	<p><i>SeqNum:GTID</i></p> <p>In the preceding syntax:</p> <ul style="list-style-type: none"><i>SeqNum</i> is the Oracle GoldenGate sequence number.<i>GTID</i> the MySQL global transaction identifier. <p>For example, if the sequence number is 00000000000000000001 and the GTID is f77024f9-f4e3-11eb-a052-0021f6e03f10:000000000000010654, then the CSN value is:</p> <p>0000000000000000000001:f77024f9-f4e3-11eb-a052-0021f6e03f10:000000000000010654</p>

Table 13-2 (Cont.) Oracle GoldenGate CSN Values Per Database

Database	CSN Value
Oracle	<i>system_change_number</i> Where: <ul style="list-style-type: none"><i>system_change_number</i> is the Oracle SCN value. Example: 6488359
SQL Server	Can be any of these, depending on how the database returns it: <ul style="list-style-type: none">Colon separated hex string (8:8:4) padded with leading zeroes and 0X prefixColon separated decimal string (10:10:5) padded with leading zeroesColon separated hex string with 0X prefix and without leading zeroesColon separated decimal string without leading zeroesDecimal string Where: <ul style="list-style-type: none">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. Examples: 0X00000d7e:0000036b:01bd 0000003454:0000000875:00445 0Xd7e:36b:1bd 3454:875:445 3454000000087500445

Connecting Microservices and Classic Architectures

This topic lists the steps to establish a connection between Oracle GoldenGate Microservices and Classic architectures.

Connect Oracle GoldenGate Classic Architecture to Microservices Architecture

Oracle GoldenGate Classic Architecture uses the data pump Extract in Admin Client and GGSCI to connect to Microservices Architecture

 **Note:**

Oracle GoldenGate Classic Architecture's pump Extract can only connect to an unsecured Microservice Architecture deployment, of which the receiver server's port is open for ingress traffic.

If the above requirement is a security concern, it is recommended to install Microservices Architecture on the same target, along with Classic Architecture, and use a reverse proxy server to allow wss distribution path between these two Microservices Architecture deployments. After this distribution path is established, the Classic Architecture deployment can pick up the trail from the same location on the target.

To connect Oracle GoldenGate Classic Architecture and Microservices follow these steps:

 **Note:**

To establish a connection between Oracle GoldenGate Classic Architecture and Microservices, only non-secured MA deployments are supported. Secure Microservices Architecture deployments are not supported.

Create a data pump Extract

 **Note:**

To perform this task, an existing data pump Extract must be running in Classic Architecture.

1. Log in to GGSCI.
2. Add a data pump Extract using the command:

```
ADD EXTRACT dp_name, EXTTRAILSOURCE ./dirdat/aa
```

This example uses, `dp_name` as the name of the data pump Extract.
3. Add the remote trail to the data pump Extract using the command:

```
ADD RMTTRAIL ab, EXTRACT dp_name, MEGABYTES 500
```
4. Edit the parameter file for the data pump Extract using the command:

```
EDIT PARAMS dp_name
```

Here is an example of the data pump Extract parameter file:

```
EXTRACT dp_name
RMTHOST hostname-or-IP-address, PORT receiver-service-port
```

```
RMTRAIL ab
PASSTHRU
TABLE pdb.schema.table;
```

Start the data pump Extract

Use the following command to start the data pump Extract `dp_name`:

```
START EXTRACT dp_name
```

Once the data pump Extract has started, the Receiver Service establishes a path and begins reading the remote trail file. The remote trail file appears in the `$OGG_VAR_HOME/lib/data` of the associated deployment running the Receiver Service.

Connect Oracle GoldenGate Microservices Architecture to Classic Architecture

To establish a connection to Classic Architecture from Microservices Architecture, the Distribution Service in Oracle GoldenGate Microservices Architecture must know where to place the remote trail file for reading.

To connect Oracle GoldenGate Microservices Architecture and Classic Architecture follow these steps:



Note:

For this procedure to work only the `ogg` protocol is supported and an existing Extract must be running in Microservices Architecture.

Task 1: Start Manager in Classic Architecture

1. Log in to GGSCI.
2. Use the command:

```
START MANAGER
```

For more information, see `START MANAGER` in *Parameters and Functions Reference for Oracle GoldenGate*.

Task 2: Add a Distribution Path

1. Launch the Distribution Service web interface.
2. Click the plus (+) sign next to **Path**. The **Add Path** page is displayed.
3. Enter the following details on the **Add Path** page:

Options	Description
Path Name	Enter the name of the Distribution Path.
Description	Enter the description of the Distribution Path.
Source	Select Extract from the drop-down list. Enter the Extract name in the text box below it.

Options	Description
Generated Source URI	Enter the location of the source trail file.
Target	<p>Select ogg as the target protocol from the drop-down list.</p> <p>Enter the following in the given order:</p> <ol style="list-style-type: none"> Target Hostname: Name of the target host service to which the connection will be established. Target Manager Port: Port number of the Oracle GoldenGate Classic Architecture Manager port. Target sub-directory for the trail file: Name of the subdirectory where the trail file is to be stored. For example, <code>.dirdat</code>. Target trail file name: Name of the target trail file, such as <code>ea</code>.
Generated Target URI	The location of the target trail file is displayed.
Target Encryption Algorithm	<p>Select NONE from the drop-down list.</p> <p>To encrypt the target trail file, select the appropriate encryption algorithm from the drop-down list.</p>
Enable Network Compression	Select this option if you want to enable network compression.
Sequence Length	Select the required value from the drop-down list for target trail sequence length. The default value is 9 .
Trail Size (MB)	Specify the value of the trail file size, as per your requirements.
Configure Trail Format	<p>Select this option if you want the trail file in any of the following formats:</p> <ul style="list-style-type: none"> TEXT SQL XML
Encryption Profile	<p>This is the encryption profile that was used to encrypt the trail file when it was generated.</p> <p>However, certain encryption methods are only available in Microservices Architecture and are not supported by Classic Architecture, so use this feature with caution.</p>

Options	Description
Target Type	Select Manager as the target type. Alternatively, you can select Collector or Receiver Service . When connecting Microservices architecture with other Microservices architecture, select the Receiver Service option. When connecting Microservices architecture with Classic architecture, select either the Manager or Collector option. If you select the Collector option, you need to start a static collector beforehand on the Classic architecture and use that static collector port as the value of the Target Manager Port field.
Begin	Select the Position in Log option from the drop-down list.
Source Sequence Number	Enter the sequence value of the source trail.
Source RBA Offset	Enter the value of the RBA offset of the source trail if you want the path to start reading from a specific RBA.

4. Click **Create Path** or **Create and Run**, as required. Select **Cancel** if you need to get out of the **Add Path** page without adding a path.

After the path is created, you'll be able to see the new path in the Distribution Service home page.