

Oracle® GoldenGate

Microservices Architecture Documentation



(26ai)
G49008-02
January 2026

ORACLE®

Oracle GoldenGate Microservices Architecture Documentation, (26ai)

G49008-02

Copyright © 2017, 2026, Oracle and/or its affiliates.

Primary Author: Preeti Shukla

Contributing Authors: Priyanka Deondi

Contributors: Alex Lima, Mack Bell, Nick Wagner, Richard Johnston, Volker Kuhr, Werner He

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	i
Documentation Accessibility	i
Related Information	i
Conventions	i

1 Overview

About Oracle GoldenGate	1
Why Do You Need Oracle GoldenGate?	1
When Do You Use Oracle GoldenGate?	2
Real-time Access to Real-time Information	3
Topologies for Oracle GoldenGate	3
Real-time Data Integration Across the Enterprise and Cloud	4
Oracle GoldenGate Product Family	5
About Oracle GoldenGate Microservices Architecture	6
Features of Oracle GoldenGate Microservices Architecture	6
Access Points for Oracle GoldenGate Microservices	7
About Deployments	9
Components of Oracle GoldenGate Microservices Architecture	10
About Service Manager	10
About Administration Server	11
About Distribution Service	11
About Receiver Server	12
About Receiver Service Path	14
About Performance Metrics Service	14
About Data Replication Components in Oracle GoldenGate	14
Types of Data Replication Configurations	14
Oracle GoldenGate Processes	15
Oracle GoldenGate Objects	16
Oracle GoldenGate 26ai: What's New	20
Directories and Variables in Microservices Architecture	20
Installation Directories and Programs in Oracle GoldenGate Microservices Architecture	20
Deployment Directories in Oracle GoldenGate Microservices Architecture	22

2 Get Started

Download Oracle GoldenGate Software	1
Verify Certification and System Requirements	1
Operating System Requirements	2
Memory Requirements	2
Oracle GoldenGate Security Privileges	3
Disk Requirements	4
Network Requirements	5
Operating System Privileges	5
Windows Console Character Sets	6
Prerequisites for Installing Oracle GoldenGate for Db2 LUW	7
Choosing an Installation System for Db2 LUW	7
Choosing and Configuring a System for Remote Capture or Delivery	7
Prerequisites for Installing Oracle GoldenGate for Db2 for i	8
General Requirements	8
Choosing an Installation Operating System	9
Supported ODBC Driver	9
Prerequisites for Installing Oracle GoldenGate for Db2 z/OS	10
Choose an Operating System for Installing Oracle GoldenGate for Db2 z/OS Remote Capture and Delivery	10
Prerequisites for Installing Oracle GoldenGate for MySQL	11
Install OpenSSL on Linux	11
Prerequisites for Installing Oracle GoldenGate for Oracle	12
Prerequisites for Installing Oracle GoldenGate for SQL Server	12
Prerequisites for Installing Oracle GoldenGate for Sybase	12
Prerequisites for Installing Oracle GoldenGate for Teradata	14
Prerequisites for Installing Oracle GoldenGate for TimesTen	14
Prerequisites for Installing Oracle GoldenGate Remote Capture and Delivery	15
Other Program and Settings	15

3 Install, Patch, and Upgrade

Installing Oracle GoldenGate	1
Installing Oracle GoldenGate Microservices Architecture	1
Performing an Interactive Installation with OUI for MA	2
Performing a Silent Installation with OUI	3
Integrating Oracle GoldenGate Microservices Architecture into a Cluster	4
Post-installation Tasks	4
Set Environment Variables for Oracle GoldenGate for Db2 z/OS	4

Patching Oracle GoldenGate	4
Downloading Patches for Oracle GoldenGate	5
Patching Using OPatch	5
Post-Patch Installation Tasks for Non-Oracle Databases	8
Post-Patching Tasks for Oracle GoldenGate for SQL Server	8
Uninstalling Patches Using OPatch	9
Upgrade	9
Obtaining the Oracle GoldenGate Distribution	9
Prerequisites	9
Oracle GoldenGate Upgrade Considerations	10
Pre-upgrade Steps	11
Upgrade Oracle GoldenGate Microservices – GUI Based	12
Upgrade Oracle GoldenGate Microservices Using REST APIs	14
Post-Upgrade Steps	16
Migrate from Classic to Microservices Architecture Using the Migration Utility	18
Prerequisites for Using the Migration Utility	18
Run the Migration Utility	20
Different Use Cases for Migration	23
Uninstalling Oracle GoldenGate Microservices Architecture	25
Removing Deployments and Service Manager	25
Removing Deployments and Service Manager Using Oracle GoldenGate Configuration Assistant	26
Using Oracle GoldenGate Configuration Assistant - Silent	26
Files to be Removed Manually	27
Uninstalling Microservices Architecture with Oracle Universal Installer	27
Uninstalling Microservices Architecture Using Silent Mode	28

4 Secure

Oracle GoldenGate Security Standards	1
Secure Deployments	1
Authentication and Authorization	2
Oracle GoldenGate Authentication and Authorization	2
Database Authentication and Authorization	6
Secure Data at Rest	6
Trail File Encryption	6
Encryption Support for Cached Files	7
Encryption Support for Persisted BR Files	7
Secure Data in Transit	7
Secure Communication Using TLS and mTLS Support	8
Oracle GoldenGate Reverse Proxy Support	11
Accountability	11

Auditing and Logging	11
Miscellaneous	12
FIPS 140-2	13
Oracle GoldenGate Security Feature: Implementation	13
Create Certificates for Secure Deployments	14
Create RootCA and Server Certificates	15
Create External Trusted RootCA and Distribution Client Certificates	16
Configure Oracle GoldenGate Reverse Proxy with NGINX	18
Prerequisites for Using ReverseProxySettings	18
Run the ReverseProxySettings Utility to Configure NGINX	19
Encrypting Trail Files	21
Using the Plugin Service for Trail File Encryption	22
Create and Apply Encryption Profile in a Deployment	22
Using Oracle Key Vault Trail File Encryption in Oracle GoldenGate	25
Using OCI KMS Trail File Encryption in Oracle GoldenGate	28
Managing Identities in a Credential Store	44
Configure Credential Store from the Admin Client	44
Specify the Alias in a Parameter File or Command	45
Encrypt and Store User Credentials	45
Configure Kerberos Authentication	46
Configure Kerberos Authentication with MA	47

5 Deploy

Add a Deployment	1
Before Adding a Deployment	1
Start the OGGCA Wizard	1
Select Service Manager Options	2
Service Manager Administrator Account	6
User Deployment	8
User Deployment Administrator	10
Summary	11
Configure Deployment	11
Finish	13
Add a Deployment to an Existing Service Manager	13
Add a Deployment in Silent Mode using OGGCA	17
Connect to the Deployment from the Web Interface	18
Start and Stop the Service Manager	18
Remove a Deployment	19
Before Removing the Deployment	19
Start OGGCA to Remove Deployment	20
Remove the Service Manager	20

Start OGGCA to Remove the Service Manager	20
Files to be Removed Manually After Removing Deployment	21

6 Manage Deployments

Add Users to a Deployment	1
Edit Users	2
Change the Administrator User Password Registered with XAG for a Deployment	3
Manage Certificates for Deployments	4
Add Client Certificate	4
Add a CA Certificate	4
Apply Certificates to an Oracle GoldenGate Deployment	4
Check Details of Certificates Used in a Deployment	5
Replace Certificates in a Deployment	6
Delegate User Management to an External Identity Provider	7
Configure IAM for Oracle GoldenGate	7
Example IAM Application and Oracle GoldenGate Authorization Profile Configured for an IAM Application	8
Configure OAM for Oracle GoldenGate	9
Example of OAM Application Configuration and Oracle GoldenGate Authorization Profile	10
Configure Microsoft Azure EntraID for Oracle GoldenGate	12
Add Root CA Certificate for Microsoft EntraID Azure	26
Create an Authorization Profile	27
Test the Authorization Profile	31
Switch Between States for a Deployment and Microservices	32
Change the State of a Deployment	33
Change the State of Microservices in a Deployment	33
Access Configuration and Log Details from the Service Manager and Microservices	34
Manage the Debug Log	37
Enable Debug Logging	37
Use the Debug Log	37

7 Prepare

Prepare Databases	1
Db2 LUW	1
Prepare Database Users and Privileges for Db2 LUW	1
Prepare Database Connection, System, and Parameter Settings	2
Prepare Tables for Processing	2
Configuring the Transaction Logs for Oracle GoldenGate	8
Db2 LUW: Supported Data Types, Objects, and Operations	10
Db2 for i	13

Prepare Database Users and Privileges for Db2 for i	14
Prepare Database Connection, System, and Parameter Settings	15
Preparing the Journals for Data Capture by Extract	15
Prepare Tables for Processing	19
Db2 for i: Supported Data Types, Objects, and Operations	21
Db2 z/OS	24
Prepare Database Users and Privileges for Db2 z/OS	25
Prepare Database Connection	26
Database Configuration	27
Prepare Tables for Processing	35
Prepare Db2 z/OS Transaction Logs for Oracle GoldenGate	37
Db2 z/OS: Supported Data Types, Objects, and Operations	40
MySQL	42
Prepare Database Users and Privileges for Oracle GoldenGate for MySQL	42
Prepare Database Connection	43
Database Configuration	44
Prepare Tables for Processing	50
MySQL: Supported Data Types, Objects, and Operations	52
Oracle	58
Prepare Database Users and Privileges for Oracle	58
Prepare Database Connection, System, and Parameter Settings	64
Ensuring Row Uniqueness in Source and Target Table	71
Transaction Log Settings and Requirements	72
Oracle: Supported Data Types, Objects, and Operations for DDL and DML	80
Oracle Autonomous AI Database	98
Prerequisites for Capturing from an Autonomous AI Database	98
Prerequisites for Applying to Oracle Autonomous Database	100
Details of Support When Using Oracle GoldenGate with Autonomous Databases	102
PostgreSQL	103
Prepare Database Users and Privileges for PostgreSQL	103
Prepare Database Connection	106
Database Configuration	111
Enabling Table-Level Supplemental Logging	113
PostgreSQL: Supported Data Types, Objects, and Operations	114
SQL Server	122
SQL Server Supported Versions	122
Prepare Database Users and Privileges for SQL Server	122
Prepare Database Connection	125
Preparing Tables for Processing	127
Prepare the Database for Oracle GoldenGate	129
CDC Capture Method Operational Considerations	133

Requirements Summary for Capture and Delivery of Databases in an Always On Availability Group	138
SQL Server: Supported Data Types, Objects, and Operations	139
Sybase	146
Prepare Database Users and Privileges for Sybase	146
Prepare Database Connection	147
Database Configuration	148
Preparing Tables for Processing	149
Sybase: Supported Data Types, Objects, and Operations	152
Teradata	157
Prepare Database Users and Privileges for Teradata	157
Prepare Database Connection	157
Preparing Tables for Processing	159
Teradata: Supported Data Types, Objects, and Operations	160
Yugabyte	163
Prepare Database Connection	163
Connecting to a FIPS-enabled Yugabyte System with Version 14 or Lower	166
Configuring SSL Support for Yugabyte	167
Yugabyte: Supported Objects and Operations	168
TimesTen	172
Prepare Database Users and Privileges for TimesTen	172
Prepare Database Connection	173
Preparing Tables for Processing	174
TimesTen: Supported Data Types, Objects, and Operations	175
Prepare Oracle GoldenGate	177
Configure Secure Database Connections from Oracle GoldenGate	177
Authenticate the Database Connection Using the Database Password Plugin	179
APIs Required for Database Password Plugin	179
Configure the Database Password Plugin to Store and Access the Database Password from a Third-Party Vault	180
Configure the Database Password Plugin to Store and Access the Database Password from OCI Vault	182
Configure the Database Password Plugin to Store and Access the Database Password from Oracle Key Vault	186
Add Database Connections	187
Add a Path Connection	189
Before Adding Extract and Replicat Processes	190
Add TRANDATA	190
Add a Checkpoint Table	192
Add Heartbeat Table	193
Create the Oracle GoldenGate CDC Cleanup Task	195
PostgreSQL: Extract Considerations for Remote Deployment	195
Prepare Oracle GoldenGate for Db2 for i	195

Update the GLOBALS File	196
Specifying Object Names	196
Adjusting the System Clock	196

8 Replicate Data

Extract	1
About Extract	1
Add an Online Extract	2
Extract Information Screen	2
Register Extract Screen (PostgreSQL)	2
Add the Extract Group	4
Downstream Capture Screen	5
Managed Options Screen	5
Extract Parameter File Screen	6
Create a Parameter File for Extract	7
Additional Parameter Options for Extract	10
Register an Extract from the Admin Client	10
Register Extract for Oracle	11
Register Extract for PostgreSQL	11
Configure Extract for Oracle Autonomous Database	13
Downstream Extract for Oracle GoldenGate Deployment	17
Configure the Source and Downstream Mining Databases	19
Configure Oracle GoldenGate Extract for Downstream Deployment	24
Configure Cascaded Downstream Extract Using Active Data Guard	25
Extract Actions	27
Access Extract Details	27
Start or Stop Extract	28
Alter Extract	28
Delete Extract	29
Positioning Extract to a Specific Start Point	30
Bounded Recovery	31
Instantiate	32
About Instantiation	32
Precise Instantiation for Db2 LUW	33
Precise Instantiation for Oracle GoldenGate for Db2 for i	33
Prerequisites for Instantiating for Db2 for i	34
Precise Instantiation Using Db2 for i Database Utilities	35
Start Extract During Instantiation	36
Positioning Extract After Startup	37
Making the Instantiation Procedure More Efficient	37
Precise Instantiation for Oracle	37

Using Oracle Data Pump for Automatic Per Table Instantiation	38
Using Recovery Manager (RMAN) for Instantiation	39
Precise Instantiation for Oracle Using Initial Load Extract	42
Precise Instantiation for MySQL to MySQL Using MySQL Shell Utilities and Oracle GoldenGate	50
Precise Instantiation for SQL Server	52
Precise Instantiation for PostgreSQL Using Initial Load Extract	54
Precise Instantiation Between PostgreSQL Environments Using pg_dump Utility	56
Monitoring and Controlling Processing After the Instantiation	59
Verifying Synchronization	60
Distribute	60
Distribution Path Streaming Protocols	60
About Distribution Service Path	61
Add a Distribution Path	62
Actions for Distribution Service Paths	67
Reposition a Path	68
Change the Path Filtering	68
Monitor the Path Information and Statistics	71
About Receiver Service Paths	72
Add Target-Initiated Distribution Paths	72
Actions for Target-Initiated Distribution Path	76
Access Network Statistics for a Path	76
About Data Streams	77
About Data Streams	77
Add Data Streams	82
Edit Data Streams Configuration	84
Access and Edit Data Streams Path Information	85
Monitor Data Streams Statistics	87
Oracle GoldenGate Data Streams REST APIs	87
Replicating Business Objects with Oracle JSON Relational Duality and GoldenGate Data Streams	87
Advantages of Using JSON Duality Views with GoldenGate Data Streams	88
Parameters Used When Configuring JSON DVs and JCTs	89
Configure GoldenGate Data Streams and JSON Relational Duality Views to Deliver Change Data	89
Replicat	90
About Replicat	90
Types of Replicat	91
About Classic or Non-Integrated Replicat	91
About Coordinated Replicat	92
About Integrated Replicat	94
About Parallel Replicat	96
Select a Replicat Type for the Deployment	99

Add a Replicat	103
Configure Replicat for Oracle Autonomous AI Database	106
Replicat Actions	108
Access Replicat Details	108
Stop, Start a Replicat	109
Alter Replicat	109
Delete Replicat	109
Configure a Replicat Parameter File	109
Additional Parameter Options for Integrated Replicat and Parallel Replicat in Integrated Mode	111
Controlling Checkpoint Retention	113
Excluding Replicat Transactions in Bidirectional Replication	113
DDL Notification on Target Tables	114

9 Administer

Data Management	1
MySQL: DDL Replication	1
MySQL: Prerequisites for DDL Configuration	2
DDL Filtering for MySQL	2
PostgreSQL: DDL Replication	5
PostgreSQL: Prerequisites for Configuring DDL	5
PostgreSQL: During DDL Replication	6
Oracle: DDL Replication	6
Overview of DDL Synchronization	6
Limitations of Oracle GoldenGate DDL Support	6
Guidelines for Configuring DDL Replication for Oracle	9
Understanding DDL Scopes	10
Correctly Identifying Unqualified Object Names in DDL	13
Enabling DDL Support	14
Filtering DDL Replication	14
Special Filter Cases	15
How Oracle GoldenGate Handles Derived Object Names	16
Using DDL String Substitution	19
Add Supplemental Log Groups Automatically	19
Removing Comments from Replicated DDL	20
Replicating an IDENTIFIED BY Password	20
How DDL is Evaluated for Processing	20
Viewing DDL Report Information	21
Using Edition-Based Redefinition	23
About Oracle GoldenGate AI Service	24
Configure and Use an AI Service	25

AI Models	26
Configure DDL Modification for Oracle GoldenGate for Sybase	26
Using Oracle GoldenGate with MySQL Group Replication	26
Oracle GoldenGate Features to Support MySQL Group Replication	27
Requirements for Supporting Group Replication	29
SSL Configuration on Group Replication Cluster	29
Procedural Replication	33
About Procedural Replication	33
Procedural Replication Process Overview	34
Determining Whether Procedural Replication Is On	34
Enabling and Disabling Supplemental Logging	35
Filtering Features for Procedural Replication	36
Handling Procedural Replication Errors	37
Listing the Procedures Supported for Oracle GoldenGate Procedural Replication	37
Monitoring Oracle GoldenGate Procedural Replication	38
Execute Commands, Stored Procedures, and Queries with SQLEXEC	39
Performing Processing with SQLEXEC	39
Using SQLEXEC	39
Apply SQLEXEC as a Standalone Statement	40
Apply SQLEXEC within a TABLE or MAP Statement	41
Using Input and Output Parameters	42
Handling SQLEXEC Errors	44
Additional SQLEXEC Guidelines	45
Mapping and Manipulating Data	46
Parameters that Control Mapping and Data Integration	46
Mapping between Dissimilar Databases	46
Globalization Considerations when Mapping Data	47
Mapping Columns Using TABLE and MAP	50
Data Type Conversions	56
Selecting and Filtering Rows	57
Retrieving Before and After Values	62
Selecting Columns	63
Selecting and Converting SQL Operations	63
Using Transaction History	64
Testing and Transforming Data	65
Using Tokens	71
Bi-Directional Replication	72
Prerequisites for Bidirectional Replication	73
MySQL: Bi-Directional Replication	77
PostgreSQL: Bi-Directional Replication	78
Preparing DBFS for an Active-Active Configuration	79
Error Management	84

Overview of Oracle GoldenGate Error Handling	84
Handling Extract Errors	84
Handling Replicat Errors during DML Operations	85
Handling Replicat Errors during DDL Operations	88
Use the Error Log	88
Automatic Conflict Detection and Resolution	89
About Automatic Conflict Detection and Resolution	89
Configuring Delta Conflict Detection and Resolution	98
Managing Automatic Conflict Detection and Resolution	101
Monitoring Automatic Conflict Detection and Resolution	105
Manual Conflict Detection and Resolution	108
Overview of the Oracle GoldenGate CDR Feature	108
Configuring the Oracle GoldenGate Parameter Files for Conflict Resolution	109
Making the Required Column Values Available to Extract	110
Viewing CDR Statistics	110
CDR Example 1: All Conflict Types with USEMAX, OVERWRITE, DISCARD	111
CDR Example 2: UPDATEROWEXISTS with USEDELTA and USEMAX	117
CDR Example 3: UPDATEROWEXISTS with USEDELTA, USEMAX, and IGNORE	119
Configuring the Oracle GoldenGate Parameter Files for Error Handling	122
Trail File Management	127
Assign Storage for Oracle GoldenGate Trails	127
Estimate Space for the Trails	127
Add a Trail	128
Change the Size of Trail Files	129
Download Trail Files	129
Delete Trail Files	130
Automate Maintenance Tasks	130
Archive Trails	132
Purge Trails	132
Purge Process	133
Using Automatic Trail File Recovery	133
Guidelines for Using Self-describing Trails	133
About Oracle GoldenGate AI Service	133
Before Adding an AI Service Provider	133
Configure and Use the AI Service	134
Admin Client Command Line Interface for Oracle GoldenGate Microservices	137
About Admin Client	137
Using Wildcards in Command Arguments	140
Using Command History	140
Storing and Calling Frequently Used Command Sequences	140
Controlling Extract and Replicat	141
Deleting Extract and Replicat	142

Creating a Parameter File Using Admin Client	143
Creating a Parameter File with a Text Editor	145
Validating a Parameter File	145
Simplifying the Creation of Parameter Files	145
Using Wildcards	146
Using OBEY	146
Using Macros	146
Using Parameter Substitution	146
Specifying Object Names in Oracle GoldenGate Input	147
Specifying Filesystem Path Names in Parameter Files on Windows Systems	147
Specifying Names that Contain Slashes	147
Specifying Case-Sensitive Database Object Names	148
Differentiating Case-Sensitive Column Names from Literals	149
Supported Database Object Names	149
Qualifying Database Object Names	150
Using Wildcards in Database Object Names	152
Performing Administrative Operations	154
Shutting Down the System	154
Changing Database Attributes	155
Changing Database Metadata	155
Adding Tables to the Oracle GoldenGate Configuration	156
Coordinating Table Attributes between Source and Target	157
Dropping and Recreating a Source Table	159
Purging Archive Logs	159
Reorganizing a DB2 Table (z/OS Platform)	160
Access the Parameter Files	160
Manage Auto Start and Auto Restart for Extract and Replicat Processes	161
Switching Replicat from Non-Integrated Mode to Integrated Mode	162
Switching Replicat from Integrated Mode to Non-Integrated Mode	163
Switching Replicat to Coordinated Mode	165
Procedure Overview	165
Performing the Switch to Coordinated Replicat	165
Administering a Coordinated Replicat Configuration	167
Performing a Planned Re-partitioning of the Workload	167
Recovering Replicat After an Unplanned Re-partitioning	168
Reprocessing From the Low Watermark with HANDLECOLLISIONS	168
Using the Auto-Saved Parameter File	169
Simplify and Automate Work with Oracle GoldenGate Macros	170
Define a Macro	170
Call a Macro	172
Call a Macro that Contains Parameters	173
Call a Macro without Input Parameters	175

Calling Other Macros from a Macro	175
Create Macro Libraries	176
Tracing Macro Expansion	177
Using User Exits to Extend Oracle GoldenGate Capabilities	178
When to Implement User Exits	178
Making Oracle GoldenGate Record Information Available to the Routine	179
Creating User Exits	179
Supporting Character-set Conversion in User Exits	180
Using Macros to Check Name Metadata	181
Describing the Character Format	181
Upgrading User Exits	183
Viewing Examples of How to Use the User Exit Functions	183

10 Monitor and Tune

Monitor	1
Purge Datastore	1
Monitor Extracts	1
Monitor Replicats	2
Monitor Microservices	3
Protocols for Performance Monitoring for Different Operating Systems	4
Recover from Datastore Corruption	5
Using Automatic Workload Repository (AWR) Reports for Oracle Database	5
Replication System Resource Usage	6
Replication Top Wait Events	6
Replication Top SQLs	7
Oracle GoldenGate Extract Performance Metrics	8
Oracle GoldenGate Integrated Replicat	9
Oracle GoldenGate Replicat	9
Monitor Oracle GoldenGate Statistics using StatsD	10
Enable StatsD using OGGCA	11
Enable StatsD with REST API Service Endpoints	13
StatsD Metrics Catalog	14
Disable StatsD using OGGCA	30
Disable StatsD with REST API Service Endpoints	30
About Integrated Diagnostics	31
How to Configure Integrated Diagnostics	32
Monitor Processes and Statistics from the Command Line	34
Monitor Oracle GoldenGate Processes, Trails, and Paths	37
Search and Read the Log Information from the Diagnosis Page	37
Search and Read Log Information for Microservices in a Deployment	39
Monitor an Extract Recovery	39

Monitor Lag	40
About Lag	40
Monitor Lag Using Automatic Heartbeat Tables	40
Reporting Lag	52
Identifying Lag in Replicat	52
Db2 z/OS: Interpret Statistics for Update Operations	53
Monitor Processing Volume	53
Use the Process Report	53
Scheduling Runtime Statistics in the Process Report	54
Viewing Record Counts in the Process Report	54
Prevent SQL Errors from Filling the Replicat Report File	54
Use the Discard File	54
Maintain Discard and Report Files	55
Parameters Used to Interpret Synchronization Lag	56
Tuning	56
Using Multiple Process Groups	56
Considerations for Using Multiple Process Groups	57
Splitting Large Tables Into Row Ranges Across Process Groups	59
Eliminating Disk I/O Bottlenecks	59
Improving I/O performance Within the System Configuration	59
Improving I/O Performance Within the Oracle GoldenGate Configuration	60
Managing Virtual Memory and Paging	60
Tuning Replicat Transactions	61
Tuning Coordination Performance Against Barrier Transactions	61
Preventing Full Table Scans in the Absence of Keys	61
Splitting Large Transactions	62
Adjusting Open Cursors	62
Improving Update Speed	62
Set a Replicat Transaction Timeout	62
Using Healthcheck Scripts to Monitor and Troubleshoot	63
Installing, Running, and Uninstalling Healthcheck Scripts	63
How to Deal with Healthcheck Information?	63
Components of Healthcheck Information	63

11 Mission Critical

Configuration Service	1
Enable the Configuration Service	2
Configuration Service: Considerations	5
Backing up the Oracle GoldenGate Environment	6

12 Quickstarts

Switching from Nonintegrated Replicat to Parallel Nonintegrated Replicat	1
Connecting Two Deployments Using External RootCA Certificate	3
Create the Distribution Client and External RootCA Certificates	4
Apply Certificates to Source and Target Deployments	6
Connect the Two Deployments Using the Distribution Path	7

13 Appendix

Sample Commands to Configure GoldenGate Data Streams for JSON Relational Duality Views	1
OGGCA Sample Template Response File for Silent Deployment	4
OGGCA Sample Template Response File for Silent Deployment	4
OGGCA Response File Example	17
Using the LogDump Utility to Access Trail File Records	32
Trail Recovery Mode	32
Trail Record Format	33
Trail File Header Record	33
Tokens Area	39
Oracle GoldenGate Operation Types	40
Checkpoint Tables Additional Details	45
Internal Checkpoint Information	47
INFO EXTRACT SHOWCH Command: Checkpoint Information	48
INFO REPLICAT, SHOWCH: Checkpoint Information	50
Oracle GoldenGate Microservices Roles	51
Supported Character Sets	58
Supported Character Sets - Oracle	59
Supported Character Sets - Non-Oracle	65
Supported Locales	73
Special Values Supported by Database Data Types	79
Commit Sequence Number (CSN)	79
Connecting Microservices and Classic Architectures	80
Connect Oracle GoldenGate Classic Architecture to Microservices Architecture	80
Connect Oracle GoldenGate Microservices Architecture to Classic Architecture	81

Preface

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

Audience

This guide is intended for developers, database users, and administrators who are responsible for installing, deploying, and configuring Oracle GoldenGate. It is assumed that readers are familiar with web technologies, low code development, and scripting. A general understanding of Windows and UNIX platforms is required.

Documentation Accessibility

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

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Related Information

[Oracle GoldenGate Documentation](#)

[Oracle GoldenGate for Distributed Applications and Analytics](#)

[Oracle GoldenGate Studio Documentation](#)

[OCI GoldenGate](#)

[Oracle Database High Availability](#)

[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.

Convention	Meaning
<i>italic</i> <i>italic</i>	Italic type indicates placeholder variables for which you supply particular values, such as in the parameter statement: TABLE <i>table_name</i> . Italic type also is used for book titles and emphasis.
monospace MONOSPACE	Monospace type indicates code components such as user exits and scripts; the names of files and database objects; URL paths; and input and output text that appears on the screen. Uppercase monospace type is generally used to represent the names of Oracle GoldenGate parameters, commands, and user-configurable functions, as well as SQL commands and keywords.
UPPERCASE	Uppercase in the regular text font indicates the name of a 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: • Extract: ext. Usage with location: ext<i>n</i>, where <i>n</i> indicates 'north' compass direction. • Replicat: rep. Usage with location: rep<i>n</i>, where <i>n</i> indicates 'north' compass direction. • Distribution Path: dp. Usage with location: dp<i>n</i>, where <i>n</i> 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

Overview

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

About Oracle GoldenGate

The Oracle GoldenGate (OGG) product family delivers real-time trusted information. Oracle GoldenGate has been a trusted provider of Data Fabric and Data Mesh capabilities built around data product thinking, decentralized architectures, event-driven pipelines, and polyglot data payloads and topologies.

Now, with Oracle GoldenGate 26ai, you will be able to leverage its transactional database replication for artificial intelligence use cases including Retrieval-Augmented Generation (RAG) and enabling real-time AI embedding. The Oracle GoldenGate platform excels at change data capture (CDC), transactional data replication, data ingestion, data pipelines for continuous transformation and loading (CTL), and a wide variety of analytics on streaming data. Oracle GoldenGate interacts with the database transaction logs to continuously capture changed data (both DML and DDL) with minimal impact to the source database. This is why Oracle GoldenGate log-based replication is preferred over batch or periodically scheduled data captures. Only through leveraging such trusted data in real-time can organizations make better, faster business decisions and actions.

Oracle GoldenGate supports a real-time hybrid, heterogeneous data fabric that connects and unifies sources and targets through a multitude of styles for data integration processing:

- Real-time changed data capture (CDC) with transactional integrity
- Improve scalability and availability of mission critical databases
- High-speed ingest into big data targets
- Real-time continuous integration (CTL)
- Bulk ETL/ELT with Data Transforms in OCI GoldenGate
- Real-time streaming analytics for data-in-motion

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.

Oracle GoldenGate's modular architecture provides flexibility in how data is extracted, transformed, and replicated across heterogeneous systems. This architecture allows for the configuration of various components, including Extract, Replicat, and Collector processes, as

well as trail files, to tailor data movement to specific needs. This modularity enables efficient handling of data at the transaction level across different platforms and topologies.

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.

Real-time Access to Real-time Information

Oracle GoldenGate software provides database high availability enabling business continuity for Oracle, MySQL, PostgreSQL, YugabyteDB, EDB Postgres Advanced Server, SQL Server, HPE NonStop, Db2 for i, LUW and z/OS. Oracle GoldenGate enables the highest service level, the Platinum Tier, of the Oracle Maximum Availability Architecture (MAA).

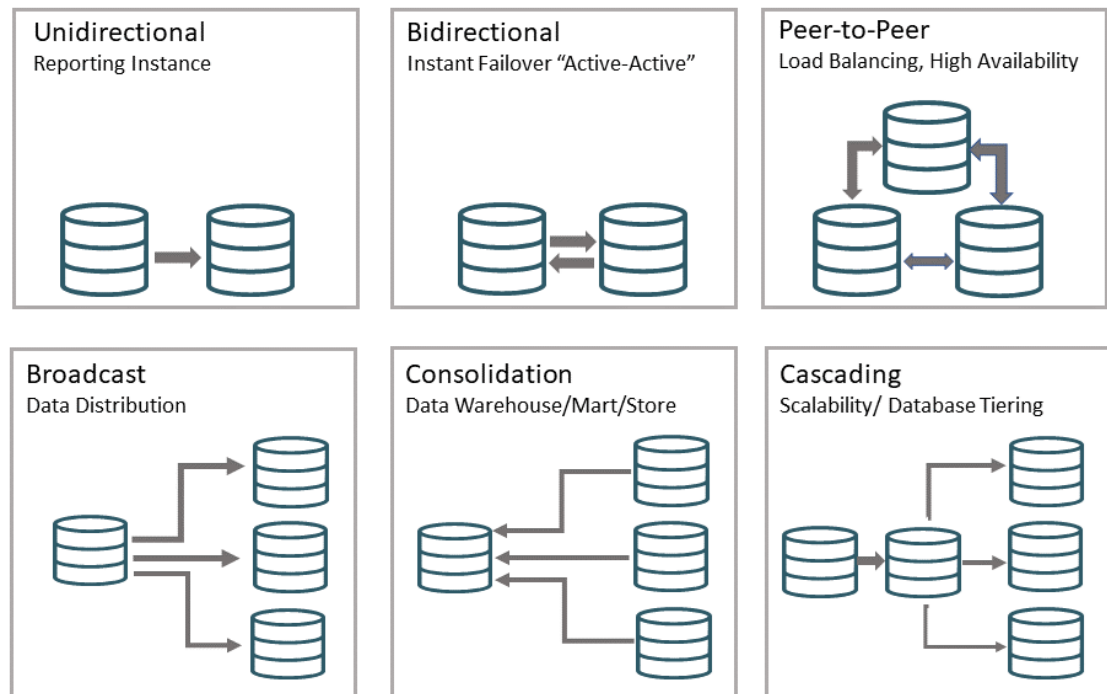
Many thousands of global banks, retailers, telecoms, healthcare companies depend on Oracle GoldenGate to reliably run their operational data platforms. Oracle GoldenGate's real-time ACID-based replication capabilities also offer an advantage for data movement for every purpose; across multivendor relational and document databases, across message stores, and across applications. Minimal source database overhead paired with high-speed processing make Oracle GoldenGate replication unique. As a result, Oracle GoldenGate includes a large collection of certified heterogeneous sources and targets and is widely used for Big Data and NoSQL data transactions and analytical payloads as well as federating data into staging and search platforms.

Oracle GoldenGate is built as a microservices/REST-based application. The modular architecture provides the flexibility for cooperative and distributed deployments across enterprise systems, on-premises and in the cloud.

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.



For different type of topology based use cases, see About OGG Solutions in the *Oracle GoldenGate Microservices Architecture Solutions*.

Real-time Data Integration Across the Enterprise and Cloud

Oracle GoldenGate captures and delivers real-time changed data to on-premises or cloud-based data warehouses, lakes and object stores, reporting systems, and other online transaction processing (OLTP) databases with minimal source system impact. Real-time data access improves business insight and analytics.

- **Oracle and non-Oracle data replication:** Oracle GoldenGate connects to many non-Oracle and open-source databases across on-premises and cloud platforms. Oracle GoldenGate also connects across multicloud relational database sources and targets such as databases in OCI, AWS Aurora and RDS environments, Azure, SQL, and Google Cloud SQL and AlloyDB. Oracle GoldenGate supports many nonrelational data platforms such as, Kafka, NoSQL databases, object storage, Hadoop, and more. See the [certification matrix](#) for a complete list of supported data platforms for each solution.
- **Real-time data warehouse:** To achieve continuous, real-time capture and delivery of changed data between operational and analytical systems.
- **Operational reporting and data integration:** Offload reporting activity from production databases to lower-cost secondary systems with current data for realtime reporting. Integrate operational data between OLTP systems in real-time.
- **Data Lakehouse ingest:** Provide continuous, real-time capture and delivery of changed data between OLTP and object storage, data lake, or data lakehouse targets. Oracle GoldenGate delivers data to Oracle or non-Oracle platforms, such as Snowflake, Azure Data Lake, Google BigQuery, and other cloud object stores.
- **Big Data integration:** Keep your entire distributed heterogeneous and multicloud data fabric and supporting applications up to date with real-time data replication, propagation, and transaction consistency. Simplify building and operating an event-based and zero data loss data fabric with five integrated enterprise-class components:

- Oracle GoldenGate connectors to capture and apply data from NoSQL and messaging systems.
- Oracle GoldenGate Distributed Applications and Analytics connectors for object storage, data lake, data warehouse targets.
- Oracle Stream Analytics for continuous ETL and data in-motion analytics.
- GoldenGate Data Streams for developer control over all GoldenGate events
- Transaction Manager for Microservices Enterprise Edition (MicroTx EE) for transactional integrity
- **Real-time streaming analytics:** Oracle Stream Analytics ingests real-time data from GoldenGate changed data, and from messaging environments like Kafka and JMS. A graphical pipeline designer can perform data transformations and apply time-series and geo-spatial analytics on these event streams. Oracle Stream Analytics is available standalone, a deployment option in OCI GoldenGate, and is included with Oracle Distributed Applications and Analytics.
- **Using GoldenGate in the cloud:** Oracle GoldenGate is available as a managed service in OCI, or as a self-managed OCI Marketplace image. Oracle GoldenGate on-premises can also be installed in any cloud on their compute instances.
- **Data Streams:** Most GoldenGate DDL and DML capabilities can also be accessed through developer APIs. Deployed using the AsyncAPI standard, developers can subscribe and publish to the GoldenGate data event model.

Oracle GoldenGate Product Family

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

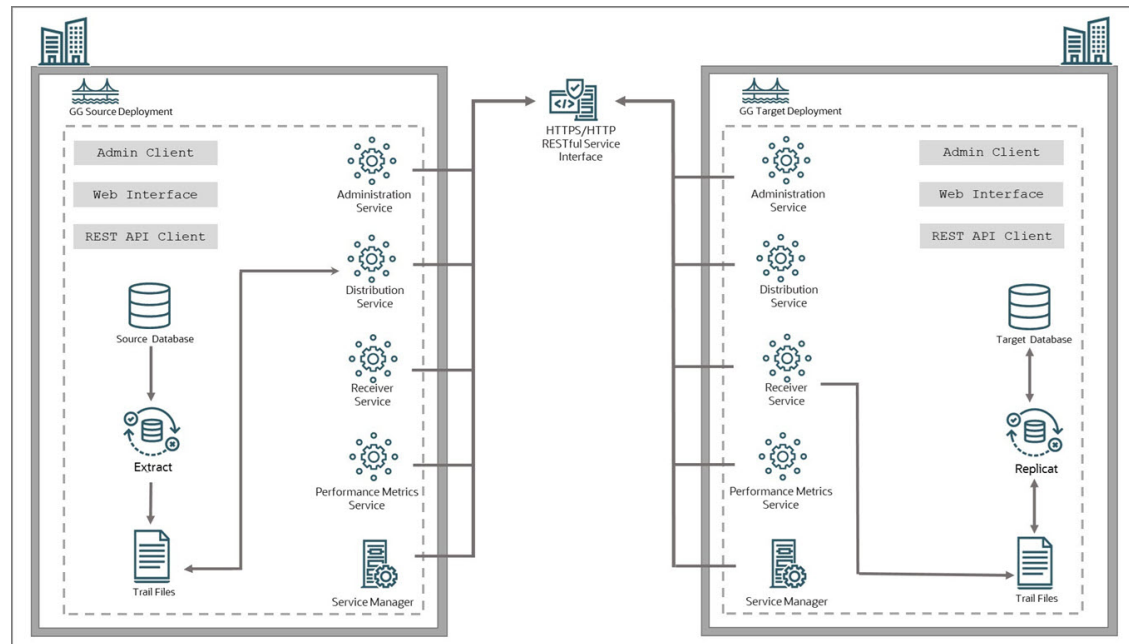
- **Oracle GoldenGate for Oracle Database:** Oracle GoldenGate Microservices for Oracle database provides all the data replication features of Oracle GoldenGate with the features of Oracle Database.
- **Oracle GoldenGate for Non-Oracle Databases:** Oracle GoldenGate Microservices for non-Oracle databases 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.

About 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.

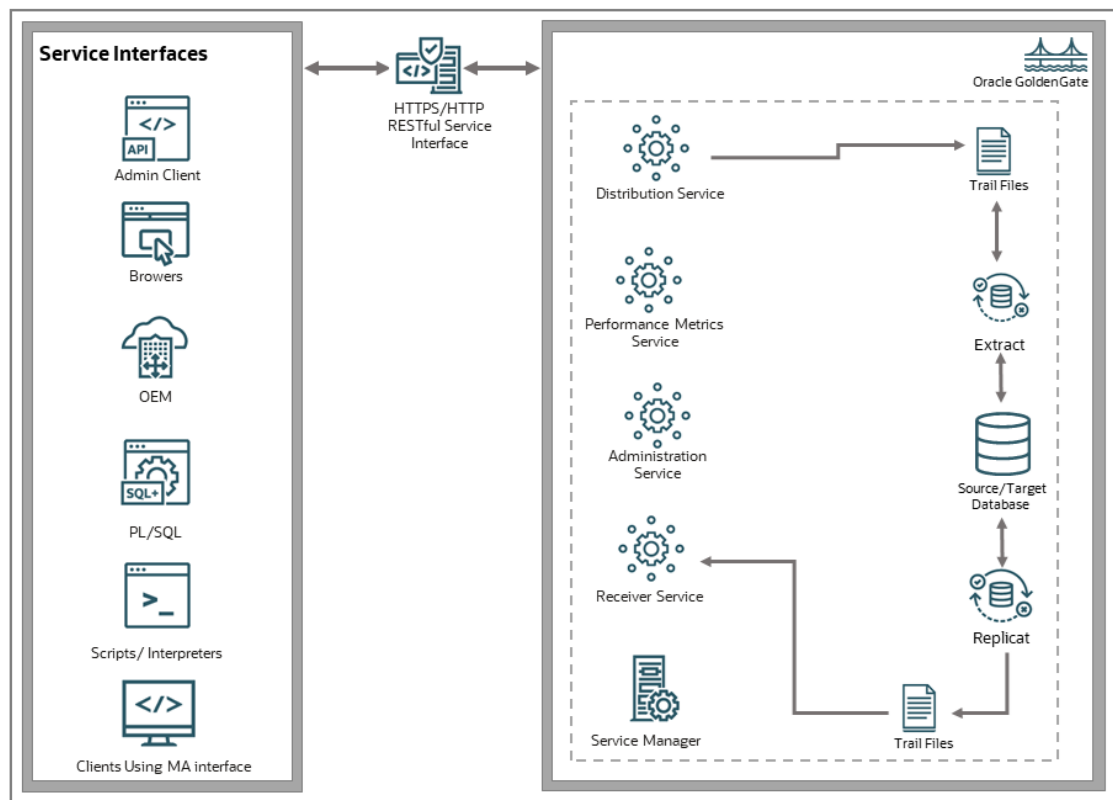
Access Points for Oracle GoldenGate Microservices

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.

Microservices Web Interface

The Oracle GoldenGate Microservices web interface provides GUI-based access to manage and work with microservices and Oracle GoldenGate processes. You can connect to the web interface from a web browser. See [Connect to the Deployment from the Web Interface](#).

From the web interface, multiple users can log into the same deployment at the same time, and each user can make changes to the environment based on their relative permissions. This means that users should ensure proper change management practices so that two users aren't trying to do things like updating the same parameter file at the same time.

An unsecure deployment uses **http** in the URL whereas a secure deployment uses **https**, where all traffic to and from the browser to the deployment is encrypted. To know more about secure and unsecure deployments, see [About Deployments](#)

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](#).

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.
- Each deployment can be managed by only one Service Manager at a time.

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 [Secure Deployments](#), 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.

Components of Oracle GoldenGate Microservices Architecture

About Service Manager

Service Manager is the primary watchdog service within Oracle GoldenGate Microservices that allows you to control and administer the deployments and associated microservices running on the host server.

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](#).

From OGGCA, you can configure the Service Manager to run in three different modes:

- Manually
- As a Daemon
- Integrated with XAG agent

Its primary functions include the following tasks:

- View and manage the status of microservices
- Manage Deployment Configuration
- Manage Service Manager Configuration
- Start and stop deployments
- Access the microservices associated with a deployment
- Access details for Administration Service, Distribution Service, Performance Metrics Service, and Receiver Service
- Add and Manage Oracle GoldenGate Users
- Add and Manage Certificates
- Add and Manage Authorization Profile for External Identity Providers
- Monitor Log Information for Diagnosing Errors
- Enable and Manage Debug Logs

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 Service

Distribution Service 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 Service 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 Service 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 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 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 Service to initiate path creation from a remote site. Paths are source-to-destination replication configurations though are not included in this release.

Note

Distribution Service 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.

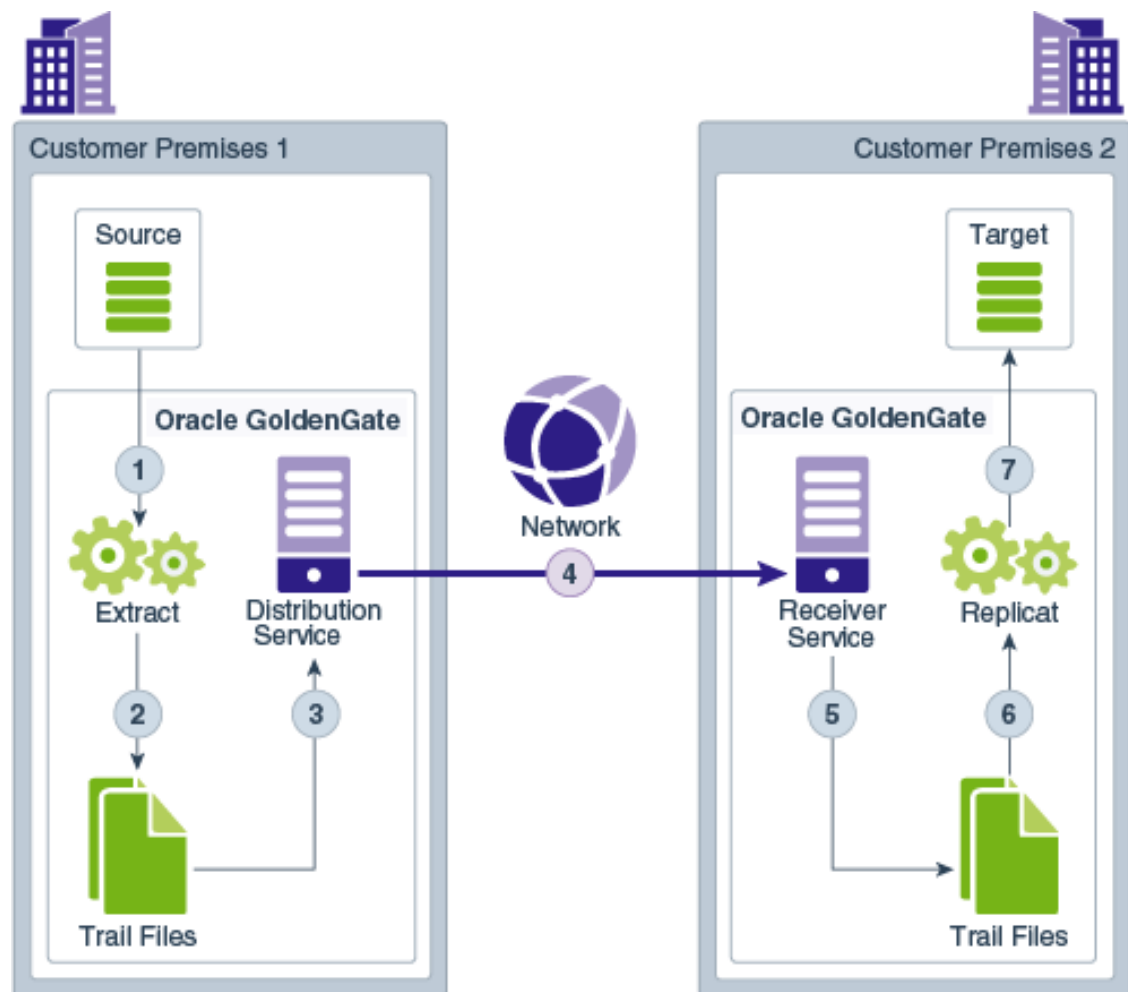
About Receiver Server

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 Receiver Service Path

Receiver Service Paths or 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 Service

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

The Performance Metrics Service uses the metrics service 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 Service 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

About Data Replication Components 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 [#unique_54](#).

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. By default, trails are stored in the sub-directory of the Oracle GoldenGate directory. You can specify nine digit sequence number for the trail file.

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. 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 setting up automated maintenance tasks to purge trail files, or delete trail files manually from the Administration Service. See [Delete Trail Files](#) and [Purge Trails](#)

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**.

Checkpoint Files and Tables

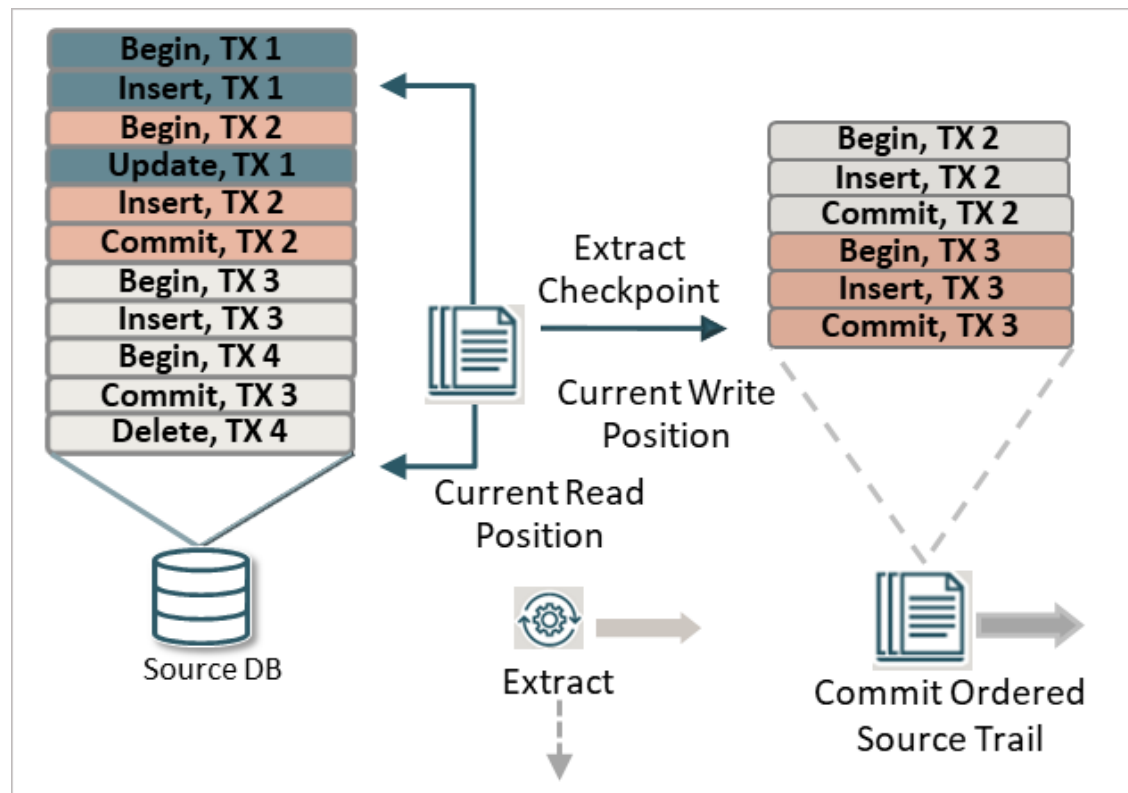
Oracle GoldenGate uses a combination of checkpoint files, checkpoint tables, and corresponding values inside the trail files to ensure that data is applied exactly once to the target platform. Oracle GoldenGate uses these objects to ensure proper replication of data, and for recovery purposes in the event of an outage or crash of either the source database, target database, or Oracle GoldenGate itself. These objects are crucial to the health of Oracle GoldenGate and should never be modified or altered without direct guidance from Oracle Support.

See [Checkpoint Tables Additional Details](#).

Checkpoint Usage by Extract, Path, and Replicat

Checkpoints are used during online change synchronization to store the current read and write position of a process. Checkpoints ensure that data changes marked for synchronization are extracted, and they prevent redundant extractions. They provide fault tolerance by preventing the loss of data if the system, the network, or an Oracle GoldenGate process needs to be restarted.

Extract, Path (Extract), and Delivery (Replicat) save positions to a checkpoint file to allow recovery in case of failure. Extract creates checkpoints for its positions in the data source and in the trail. Replicat creates checkpoints for its position in the trail. The following diagram demonstrates this workflow:



Checkpoint information is maintained in checkpoint files in the default Oracle GoldenGate directory: \$OGG_VAR_HOME/lib/checkpt

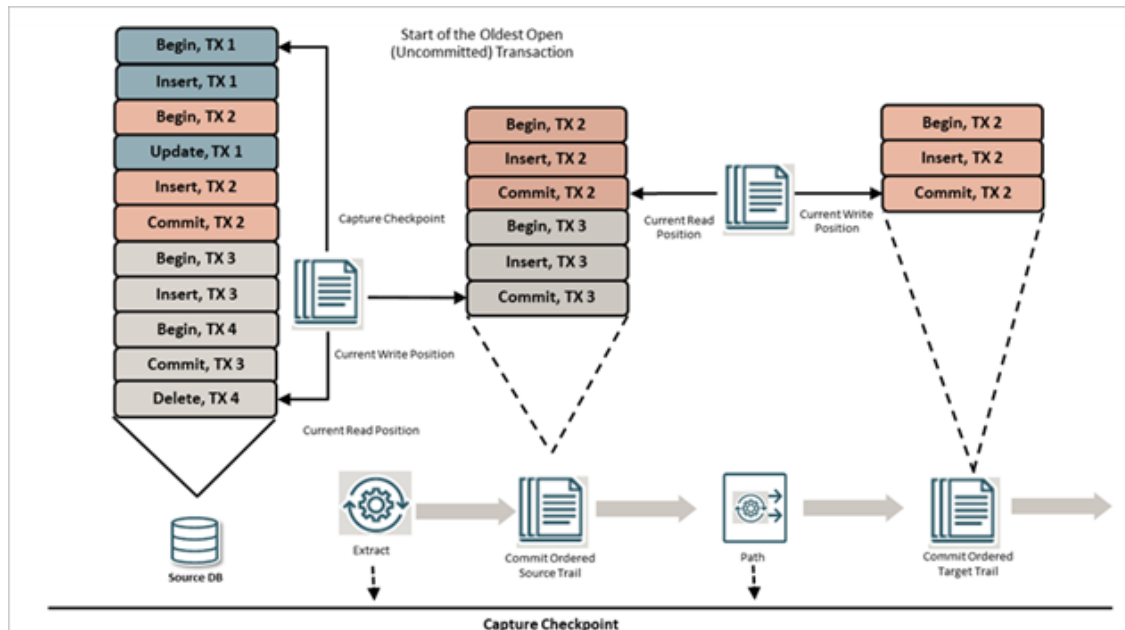
This is not to be confused with the optional checkpoint table stored on the database. Extract places a write checkpoint, known as the current checkpoint, in the trail. The current checkpoint is the position in the trail where Extract is currently writing.

Checkpoints store the current read/write position of a process. They prevent redundant extractions and provide fault tolerance.

Oracle GoldenGate uses the checkpoints for information purposes only, as the source of truth is within the trail files.

Path Checkpoint

The Distribution Service Path checkpoint adds an extra layer of reliability for Microservices Architecture when transferring trail files between deployments. The following image describes the DISTPATH checkpoint mechanism used in deployments.



The current Read is the checkpoint of what was last read. The checkpoint checks for changes every second. This frequency is configurable. The Write checkpoint identifies what was last written to the trail. Oracle GoldenGate keeps track of the writing position in the trail.

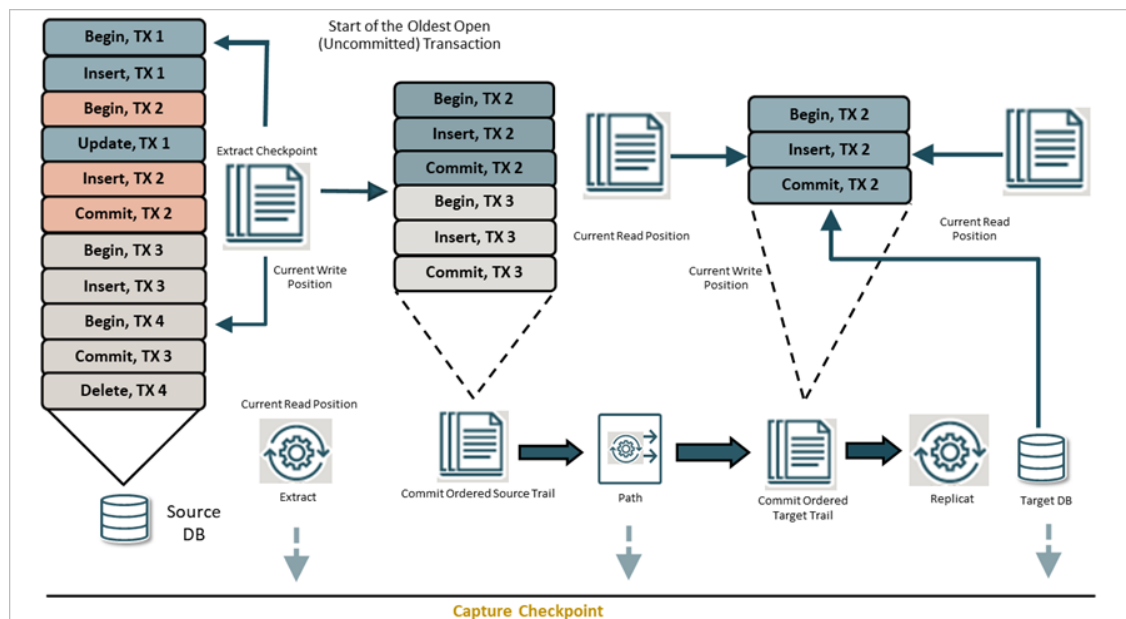
The Distribution Service Path are not aware of any open transactions because they only process the records that have been committed.

Replicat Checkpoint

Replicat makes checkpoints in the checkpoint table to mark its last read and apply position.

To view process checkpoints, use the `INFO REPLICAT` command with the `SHOWCH` option. The basic command shows current checkpoints.

The current checkpoint is the position of the last record read by Replicat in the trail and applied to the database. The source of truth for applied transactions is the checkpoint table. The checkpoint file used by Replicat and `INFO` commands are approximate and advisory information only. See the following image to understand this workflow:



Oracle GoldenGate 26ai: What's New

Oracle GoldenGate 26ai includes various new features, which significantly enhance its capabilities, making it more scalable and highly available. Review the list of new features, enhancements, and behavior changes in Oracle GoldenGate 26ai.

To see a list of new features introduced in Oracle GoldenGate 26ai, see [Release 26ai - New Features](#).

For a list of enhancements introduced in Oracle GoldenGate 26ai, see [Release 26ai - Enhancements](#).

For behavior changes in Oracle GoldenGate 26ai, see [Default Release 26ai - January 2026](#).

For deprecated and desupported features, see [Deprecated and Desupported Features](#).

Directories and Variables in Microservices Architecture

The topic describes the directories and utilities that are installed and deployed in Oracle GoldenGate Microservices Architecture.

Installation Directories and Programs in Oracle GoldenGate Microservices Architecture

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

Table 1-1 Directories in an Oracle GoldenGate MA installation

Directory	Description
bin	Sub-directory for most of the Oracle GoldenGate executable files.

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

Directory	Description
deinstall	Location of deinstall.sh, which is the software deinstallation script.
jdk	Java Developer Kit directory
lib	Contains libraries, utility files, and scripts.
oui	Oracle Universal Installer directory
OPatch	Location of Oracle Patch Utility directory to install patches (opatch).

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

Name	Description	Default Directory
adminclnt	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

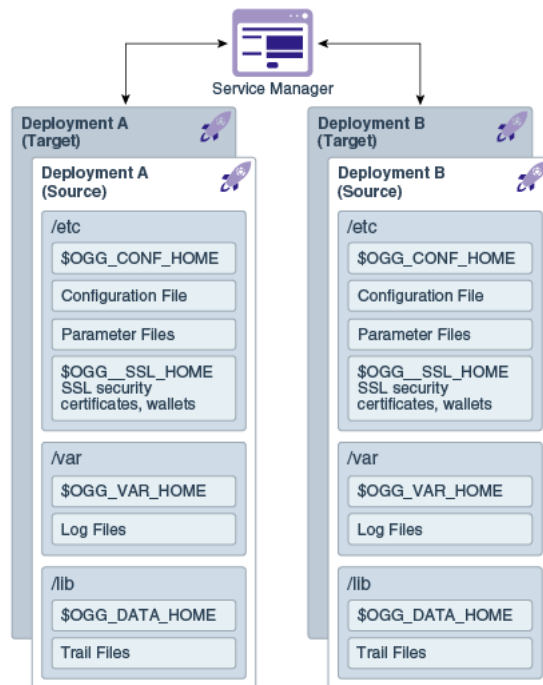
Name	Description	Default Directory
pmsrvr	The Performance Metrics Service uses the metrics service to collect and store instance deployment performance results.	\$OGG_HOME/bin
recvsrvr	A Receiver Service is the central control service that handles all incoming trail files.	\$OGG_HOME/bin
replicat	Replicat 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	A 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

Deployment Directories in Oracle GoldenGate 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.

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



The following table describes the key MA directories and the variables that are used when referring to those directories during an Oracle GoldenGate deployment configuration. Only those parameters can be edited that have not been marked as Read-only in the following table. 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 (Read-only)	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 (Read-only)	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

Directory Name	Variable	Description	Default Directory Path
Deployment variable home (Read-only)	OGG_VAR_HOME	The location where each deployment logging and reporting processing artifacts are stored.	/ogg_deployment_location/var
Deployment data home (Read-only)	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.

Oracle GoldenGate Security Features

This section presents the Oracle GoldenGate Security features available with the current release. The following table lists the security aspect and the associated Oracle GoldenGate feature that implements it.

Support ed	Oracle GoldenGate Security Feature	Description	R e f e r e n c e
✓	Secure Administration	• All administrative operations are authorized and made with REST API calls • Support for TLS 1.3 (default) and TLS 1.2	Secure Deployment and Configuration

Support ed	Oracle GoldenGate Security Feature	Description	R e f e r e n c e p o r t

Support ed	Oracle GoldenGate Security Feature	Description	R e f e r e n c e
✓	Authentication	• Credentials (user name/password) • Client certificates (without Reverse Proxy) • Single Sign On (SSO) support with external Identity Providers using OAuth/OIDC • OCI: Oracle Identity Manager (IAM), Oracle Identity Cloud Service (IDCS) • On-Premise: Oracle Access Manager (OAM) • Token-based Authentication	Authenticating and Authorizing Connections to the Cloud

Support ed	Oracle GoldenGate Security Feature	Description	R e f e r e n c e u r e D e p l o y m e n t s

Support ed	Oracle GoldenGate Security Feature	Description	R e f e r e n c e
✓	Password management and MFA	- Local password policy based on character length and rules - Supported and enforced by external Identity Provider (IDP)	S e r v i c e M a n a g e r A d m i n i s t r a t o r A c c o u n t D e l e g a t e U s e r M a n a g e m e

Support ed	Oracle GoldenGate Security Feature	Description	R e f e r e n c e
			n t t o a n E x t e r n a l I d e n t i t y P r o v i d e r

Support ed	Oracle GoldenGate Security Feature	Description	R e f e r e n c e
✓	Role based Access Control (RBAC)	Hierarchical Role based layout: • Security: User with this privilege, can perform all administrative tasks and authorizing new clients. • Administrator: User with this privilege, can perform all administrative tasks, but no authorizing new clients. • User: User with this privilege, can retrieve only status or monitoring information. • Operator: User with this privilege, with operator role can start and stop processes.	S e r v i c e M a n a g e r A d m i n i s t r a t o r A c c o u n t

Support ed	Oracle GoldenGate Security Feature	Description	R e f e r e n c e
✓	Database Credentials	• Credentials stored in secure PKCS#12 wallet • Kerberos support, if available for the database, for example: Oracle.	C o n f i g u r e K e r b e r o s A u t h e n t i c a t i o n w i t h M A

Support ed	Oracle GoldenGate Security Feature	Description	R e f e r e n c e
✓	Database Connectivity	Support based on Database client capabilities Example: Oracle using TCPS or Native Network Encryption.	A d d i t i o n s

Support ed	Oracle GoldenGate Security Feature	Description	R e f e r e n c e
✓	Network Trail File Distribution (Data in Transit)	- Secured with industry standard secure streaming Websocket protocol (WSS) - Support for mutual TLS using client certificate	D i s t r i b u t i o n P a t h S t r e a m i n g P r o t o c o l S e c u r e C o m m u n i c a t i o n

Support ed	Oracle GoldenGate Security Feature	Description	R e f e r e n c e
			S i n g l e s a n d m u l t i p l e s u p p o r t
	Proxy/DMZ	• Support for reverse and forward Proxy • Target-initiated Distribution Path for DMZ systems	A b o u t D i s t r i b u t i o n S e r v i c e P a t h

Support ed	Oracle GoldenGate Security Feature	Description	R e f e r e n c e
✓	Trail File Encryption (Data in Rest)	• Encryption support using AES128, AES192 or AES256 • Support for Master key from external Key Management System (OKV, OCI-KMS) • Support of multiple Master Keys	Trail File Encryption
✓	Auditing	• Logging of REST API Calls • Enabled System Security Logging	Auditing and Logging

2

Get Started

Start your data replication journey with downloading Oracle GoldenGate, verifying certificates, understanding server and database requirements, understanding the databases-specific prerequisites before installing Oracle GoldenGate, and preparing the server for installing Oracle GoldenGate.

Download Oracle GoldenGate Software

You can download Oracle GoldenGate from the Oracle Software Delivery Cloud site at <https://edelivery.oracle.com/osdc/faces/SoftwareDelivery>. This site hosts the first Generally Available (GA) releases of Oracle GoldenGate software versions and there may be a more recent patch version available on <https://support.oracle.com>.

It is extremely important to ensure that your Oracle GoldenGate installation is running the most recent release of the Oracle GoldenGate software. Oracle always recommends running the latest bundle patch for all Oracle GoldenGate products and this is crucial if you are installing Oracle GoldenGate for a new project or if you are updating an existing environment to the latest bundle patch.

For Oracle GoldenGate Microservices Architecture installations, patches require the GA release to be installed first, followed by the patch to apply, unless there is a patch listed with a description of 'Complete Install' in which case that patch release can be installed directly without having to first install the GA release version.

Refer to the blog [How to obtain the latest Oracle GoldenGate Microservices Architecture bundle patches](#) for a list of latest bundle patches and links to the respective MOS patches for the required build.

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

Learn about the operating system resources required to install and run Oracle GoldenGate.

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*.

Db2 z/OS: Memory Requirements

Oracle GoldenGate requires the following memory resources on the Oracle GoldenGate remote system and the database host system. For the memory resources required on a remote system, see the [All Platforms](#) section above.

On the Db2 host system

When setting up the Workload Manager (WLM) environment for the Extract log read components, it is recommended to set NUMTCB in the range of 10-40 depending on your environment. Refer to the IBM documentation for more information.

Oracle GoldenGate Security Privileges

This section outlines the security privileges that Oracle GoldenGate requires on a source system and on a Windows or Linux target system.

The person who installs Oracle GoldenGate must have read and write privileges on the Oracle GoldenGate installation directory, because steps will be performed to create some sub-folders and run some programs.

Service Manager, Replicat, and Receiver Service (program name is *server*) are active. The Service Manager controls the other processes and interacts with Receiver Service to receive incoming data, while Replicat applies data to the target database through ODBC.

It is recommended that Oracle GoldenGate processes should be assigned a user account that is dedicated to Oracle GoldenGate and cannot be used by any other program. One user account can be used by all of the Oracle GoldenGate processes. This account must have privileges to read, write, and delete files and directories within the Oracle GoldenGate installation directory.

If the Extract user profile does not have the required authority, Extract will log the following errors and stop.

```
[SC=-1224:SQL1224N A database agent could not be started to service a request, or was terminated as a result of a database system shutdown or a force command.SQL STATE 55032: The CONNECT statement is invalid, because the database manager was stopped after this application was started]
```

The user profile must be specified with the USERIDALIAS parameter when you configure the parameter files and in the DBLOGIN command prior to issuing any Admin Client commands that interact with the database.

Disk Requirements

This section lists the disk space requirements for Oracle GoldenGate and specific disk space considerations for Db2 z/OS database.

Disk Requirements for Oracle GoldenGate Installation Files

The disk space requirements for an 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 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 affected 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 Requirements

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 to use a reverse proxy. Oracle GoldenGate Microservices includes a script called ReverseProxySettings that generates configuration file for only the NGINX reverse proxy server.
See [Configure Oracle GoldenGate Reverse Proxy with NGINX](#).
- 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.
- 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.
- Configure your firewalls to accept connections through the Oracle GoldenGate ports.

Operating System Privileges

Following are the operating system privileges required to install Oracle GoldenGate Microservices Architecture, create Deployments, and run Oracle GoldenGate processes:

General Privileges

- The user who installs Oracle GoldenGate must be granted read and write privileges on the Oracle GoldenGate software home directory.
- The user who configures deployments using the oggca.sh script and creates Oracle GoldenGate Extract and Replicat processes must have read, write, and delete privileges on files and subdirectories in the Oracle GoldenGate directory.
- For Extract processes that read from transaction logs and backups, the user must have read access to the logs and backup files.

Windows Specific Privileges

- The user installing Oracle GoldenGate and creating deployments must be a member of the server's Administrators group.
- The same user account must be used to create all deployments, when multiple deployments are needed.
- If configuring the Service Manager to run as a service on Windows:
 - The Service Manager service must initially be registered to run as the user account that creates the deployment.
 - Grant the user the '**Log on as a service**' right prior to registering the service.
- To change the Service Manager service to run as another user after initial registration:
 - The user account that is to be used to run the Oracle GoldenGate Service Manager service must be a member of the server's Administrators group and must have been granted the '**Log on as a service**' right.
 - Stop all GoldenGate processes and deployments.
 - Stop the Oracle GoldenGate Service Manager service.
 - Grant the user account **Full control** to the ewallet.p12 files located in the var\lib\credential\secureStore directory of each deployment, including the Service Manager deployment.
 - Edit the '**Log on as**' property of the **Oracle GoldenGate Service Manager** service using the Windows services applet and change the account to the new user.
 - Start the service, deployments, and any Extract and Replicat processes.

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 the Admin Client command output. So, a non-matching console character set causes characters not to display correctly. You can set the character set of the console before opening an Admin Client 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).

Prerequisites for Installing Oracle GoldenGate for Db2 LUW

Learn the prerequisites for installing Oracle GoldenGate for a Db2 LUW database.

Choosing an Installation System for Db2 LUW

To install Oracle GoldenGate for Db2 LUW, you can use either of the following configurations:

1. To connect Oracle GoldenGate remotely, choose from the following options to connect to Db2 LUW:
 - Db2 Connect v10.5 or greater
 - IBM Data Server Client v10.5 or greater
 - IBM Data Server Runtime Client v10.5 or greater

If Oracle GoldenGate does not connect remotely to the Db2 LUW server, refer to point 3.

2. For remote capture/apply, catalog the remote database server to configure Oracle GoldenGate to connect remotely to the database server. See [Choosing and Configuring a System for Remote Capture or Delivery](#).
3. Install Oracle GoldenGate, and provide the values for the following environment variables during installation:
 - LD_LIBRARY_PATH should have the path to installed client/server lib64 directory.
 - DB2INSTANCE should be set to the value of the local instance on the server where Oracle GoldenGate will be running.

Choosing and Configuring a System for Remote Capture or Delivery

In a remote installation, you install Oracle GoldenGate on a server that is remote from the source or target database server. This server can be any Linux, UNIX, or Windows platform that Oracle GoldenGate supports for the Db2 LUW database. The Oracle GoldenGate build must match the version of Db2 LUW that is running on the installation server.

In this configuration, the location of the database is transparent to Extract and Replicat. Extract can read the Db2 logs on a source Db2 LUW database server, and Replicat can apply data to a target Db2 LUW server.

To Configure Remote Capture or Delivery:

1. Install and run Db2 for LUW on the remote server that has DB2 Connect.
2. Catalog the remote server in the Db2 source or target database by using the following DB2 command.

```
catalog tcpip node db2_node_name remote remote_DNS_name
```

3. Catalog the Db2 target node in the Db2 LUW database on the remote server by using the following DB2 command:

```
catalog tcpip node db2_node_name remote remote_DNS_name server  
remote_port_number
```

4. Add the Db2 source or target database to the Db2 catalog on the remote server by using the following DB2 command:

```
catalog db database_name as database_alias at node db_node_name
```

Note

Refer to the IBM Db2 LUW documentation for more information about these commands.

5. Download and install the Oracle GoldenGate build that is appropriate for the Db2 LUW database on the remote server.

Prerequisites for Installing Oracle GoldenGate for Db2 for i

Learn the prerequisites for installing Oracle GoldenGate for a Db2 for i database.

General Requirements

- Portable Application Solution Environment (PASE) must be installed on the system.
- Java 8 must be installed on the IBM i and the Linux host system where GoldenGate for IBM i will run.
- OpenSSH is recommended to be installed on the system. OpenSSH is part of the IBM Portable Utilities licensed program and allows SSH terminal access to the system in the same manner as other Linux system.
- A library/schema should be dedicated to each deployment of Oracle GoldenGate on the IBM i system. The value of the library/schema is taken from the Replication Schedma property.
- The IBM Db2 for i Program temporary fixes (PTFs) that are required by release for Oracle GoldenGate are detailed in the following tables:

IBM i7.3 Group PTF	Level	Name	Notes
SF99730	23257	Cumulative PTF	NA
SF99731	12	All PTF groups except cumulative PTF package	.
IBM i7.4 Group PTF	Level	Name	Notes
SF99740	23313	Cumulative PTF	NA
SF99741	9	All PTF groups except cumulative PTF package	NA
IBM i7.5 Group PTF	Level	Name	Notes
SF99750	23306	Cumulative PTF	NA

IBM i7.5 Group PTF	Level	Name	Notes
SF99751	5	All PTF groups except cumulative PTF package	NA

These required PTFs are the levels at which Oracle GoldenGate has been tested against for the 23ai releases. To check the group PTF levels, you must use the WRKPTFGRP command from a 5250 terminal session and check for the specific PTFs with the commands shown in the preceding tables. The specific extra PTFs must be at least temporarily applied.

Choosing an Installation Operating System

Oracle GoldenGate for Db2 for i operates remotely on Intel Linux systems. A few components will be automatically copied to the IBM i systems.

Oracle GoldenGate for Db2 for i supports the IBM i Access ODBC Driver 64-bit. For more information, see [Supported ODBC Driver](#) and [ODBC Driver for IBM i Access Client Solutions](#).

Consider the following:

- The best performance is seen with a system that has the lowest network latency to the IBM i system that you use. Although it is possible to run over a wide area network, the performance suffers due to the increased network latency.
- No special requirements beyond what capture already has for Oracle GoldenGate delivery. Because this Oracle GoldenGate release is a fully-remote distribution, the former Oracle GoldenGate Db2 for i remote product is no longer shipped separately. However, Windows is not supported in Oracle GoldenGate for Db2 for i in this release. If you still require delivery to Db2 for i from Windows, then Oracle GoldenGate Db2 for i remote 12.3 is still available.

Supported ODBC Driver

Starting from Oracle GoldenGate release 21.12, Oracle GoldenGate for Db2 for i supports the IBM i Access ODBC Driver 64-bit version 1.1.0.27 or higher.

See [Linux, macOS, and PASE Application Packages](#) for more information on IBM i Access ODBC Driver and follow the steps provided to install IBM i Access application package for the Linux operating system.

In addition, install unixODBC driver manager to be used with the Db2 ODBC driver on all supported Linux operating systems. For example, to install the unixODBC driver manager on the Red-Hat Linux operating system, use the yum command. See [Installing the unixODBC driver manager](#) for more information.

After the IBM i Access ODBC Driver and the unixODBC driver manager are installed successfully, add appropriate values in the `odbcinst.ini` and `odbc.ini` files to register the driver and the system.

You can test if the DSN and drivers are configured properly by testing the connection using the `isql` command. For example:

```
isql -v DSN user password
```

Prerequisites for Installing Oracle GoldenGate for Db2 z/OS

Learn the prerequisites for installing Oracle GoldenGate for a Db2 z/OS database.

Choose an Operating System for Installing Oracle GoldenGate for Db2 z/OS Remote Capture and Delivery

Oracle GoldenGate for Db2 z/OS operates remotely on zLinux, AIX or Intel Linux systems. To capture data, a component must be installed on the Db2 z/OS system before installing Oracle GoldenGate. This component contains the Db2 instance to allow Oracle GoldenGate to read the Db2 log data.

To install Oracle GoldenGate on a remote zLinux, AIX or Linux system, choose from the following options to connect to Db2 z/OS:

- IBM Data Server Driver for ODBC and CLI v11.5 or greater
- IBM Data Server Client v11.5 or greater
- IBM Data Server Runtime Client v11.5 or greater
- Db2 Connect v11.5 or greater

Consider the following:

- Extract uses the Db2 driver to read the transaction logs from the Db2 subsystem on the z/OS system. If one of the other drivers is not already installed, the IBM Data Server Driver for ODBC and CLI is the most lightweight and easy to configure driver and is recommended for most configurations, although the other drivers are also suitable.
- The driver requires a valid Db2 Connect license suitable for connecting to a Db2 z/OS database, although the use of the Db2 Connect product is not required to be used.
- To capture Db2 log data, the log reader component must be installed in a Library (PDSE) on the z/OS system. Load Libraries (PDS) are not supported. The library must be authorized program facility (APF) helps your installation protect the system. APF-authorized programs can access system facility (APF) authorized. The log read component is called through SQL from the remote system and since it is APF authorized, an authorized Workload Manager (WLM) environment must also be used to run these programs since the default Db2 supplied WLM environment is not able to run authorized workload.
- No special requirements beyond what capture already has for Oracle GoldenGate delivery. Because this Oracle GoldenGate release is a fully-remote distribution, the former Oracle GoldenGate Db2 Remote product is no longer shipped separately. However, Windows is not supported in Oracle GoldenGate for Db2 z/OS in this release. If you still require delivery to z/OS from Windows, then Oracle GoldenGate Db2 Remote 12.2 is still available.
- UNIX System Services (USS) is no longer required (as in prior releases) except for a few installation procedures.
- Windows only: To apply data to a Db2 target from Windows, Oracle GoldenGate Db2 Remote v12.2 must be used. Capture is not supported in this scenario.
- Install Oracle GoldenGate Db2 Remote on a remote system for remote delivery to the Db2 target system. In this configuration, Replicat connects to the target Db2 database by using

the ODBC API that is supplied in DB2Connect. This configuration requires Db2 z/OS to be installed on the remote system.

Note

All of the Oracle GoldenGate functionality that is supported for Db2 z/OS is supported by DB2Connect. In addition, ASCII character data is converted to EBCDIC automatically by DB2Connect.

- Although it is possible to install Oracle GoldenGate on zLinux, AIX, and Intel based Linux, the best performance is seen with a system that has the lowest network latency to the z/OS system that you use. Although it is possible to run over a wide area network, the performance suffers due to the increased network latency. Oracle recommends using a zLinux partition on the same physical hardware as the z/OS system that is running Db2 using Hipersockets or a VLAN between the partitions. Otherwise, systems connected with OSA adapters in the same machine room, would be the next best choice. Alternatively, the fastest Ethernet connection between the systems that is available would be acceptable.

Using the Remote Delivery to the Db2 z/OS using DB2Connect

- For the intermediary system, select any platform that Oracle GoldenGate supports for the Db2 for LUW database. This is the system on which Oracle GoldenGate is installed.
- Install and run Db2 z/OS on the selected remote system so that the Replicat process can use the supplied DB2Connect driver.
- Catalog the Db2 target node in the Db2 z/OS database on the remote system by using the following Db2 command:

```
catalog tcpip node db2_node_name remote DNS_name server DB2_port-number
```

- Add the target Db2 database to the Db2 z/OS catalog on the intermediary system by using the following Db2 command:

```
catalog db database_name as database_alias at node db_node_name
```

See the IBM Db2 z/OS documentation for more information about these commands.

Prerequisites for Installing Oracle GoldenGate for MySQL

Learn the prerequisites for installing Oracle GoldenGate for a MySQL database.

Install OpenSSL on Linux

Oracle GoldenGate 26ai for MySQL requires that OpenSSL 1.1 are present on the Oracle GoldenGate server prior to creating a deployment.

OpenSSL 1.1 is included with the core operating system packages of OEL 8 and RHEL8 but is not included with OEL 9 or RHEL 9. So, you need to manually install it for these operating systems/versions.

To install the required OpenSSL 1.1 libraries (`libssl.so.1.1` and `libcrypto.1.1`), download and install the MySQL Connector/ODBC, version 8.0.33, using the following instructions:

- Select version **8.0.33** from the **Product Version** drop-down menu, using the link, <https://downloads.mysql.com/archives/c-odbc/>.

2. Select **Linux-Generic** from the **Operating System** drop-down menu.
3. Click the **Download** link for the **x86, 64-bit** version of the package: **Linux - Generic (glibc 2.28) (x86, 64-bit), Compressed TAR Archive**.
4. Save the file to a location of your choice, such as /opt/app/, and untar the file, as shown in the following example:

```
tar -xvzf mysql-connector-odbc-8.0.33-linux-glibc2.28-x86-64bit
```

5. The OpenSSL 1.1 libraries needed for Oracle GoldenGate are located in the mysql-connector-odbc-8.0.33-linux-glibc2.28-x86-64bit/lib/private folder. Add this path to the LD_LIBRARY_PATH system variable, prior to creating a MySQL deployment, as shown in the following example.

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/app/mysql-connector-odbc-8.0.33-linux-glibc2.28-x86-64bit/lib/private/
```

Prerequisites for Installing Oracle GoldenGate for Oracle

Learn the prerequisites for installing Oracle GoldenGate for Oracle database.

Disk space is 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*.

Prerequisites for Installing Oracle GoldenGate for SQL Server

Learn the prerequisites for installing Oracle GoldenGate for a SQL Server database.

Creating an Oracle GoldenGate for SQL Server deployment on Linux requires that the unixODBC driver manager and unixODBC-devel packages be installed before launching the Oracle GoldenGate Configuration Assistant (OGGCA).

For Linux, run the following command:

```
sudo yum install unixODBC
```

```
sudo yum install unixODBC-devel
```

Prerequisites for Installing Oracle GoldenGate for Sybase

Learn the prerequisites for installing Oracle GoldenGate for a Sybase database.

Creating an Oracle GoldenGate for Sybase deployment requires that a supported operating system version of the **SDK FOR SAP ASE 16.0.3** and the Linux Network Libraries (libnsl) package be installed before launching the Oracle GoldenGate Configuration Assistant (OGGCA).

Required Linux Packages

The Linux Network Libraries (libnsl) package is required prior to creating a deployment. Run the following command to install the package:

```
sudo yum install libnsl
```

Sybase SDK FOR SAP ASE Installation

The following are basic steps for installing the SDK FOR SAP ASE 16.0.3.

For detailed instructions and requirements, refer to the [SAP documentation](#).

1. Verify which operating systems are supported for this version of Oracle GoldenGate for Sybase by reviewing the certification matrix [Certification Matrix](#).
2. Download the SDK FOR SAP ASE 16.0.3 available at <https://me.sap.com/softwarecenter>, and select the download based on the operating system where Oracle GoldenGate for Sybase is to be installed.

Note

Do not install the SDK FOR SAP ASE 16.0 version, as this is known to cause issues for the Oracle GoldenGate Administration Service.

3. Run the following command to untar the downloaded package to a staging folder and then navigate to the extracted folder.

```
tar -xvf SDKASE160003PL_15-81009139.TGZ-C /path/staging  
cd /path/staging/ebf30977
```

4. Launch the setup.bin program and follow the prompts to select the installation folder.

```
./setup.bin
```

5. When choosing which **Install Set** to install, select **Custom** and at a minimum, install the **Open Client>DB-Library** package.
6. After the installation is complete, navigate to the Sybase SDK installation directory, as the user that will install Oracle GoldenGate.

Example:

```
[oracle@localhost]$ cd /opt/sap
```

7. Source the Sybase SDK environment variables. This sets the correct values for the LD_LIBRARY_PATH and SYBASE session variables, which are required to create an Oracle GoldenGate for Sybase deployment.

```
[oracle@localhost sap]$ . ./SYBASE.sh
```

8. Run the following commands to confirm that both variables are set correctly and then proceed to create the deployment.

```
[oracle@localhost sap]$ echo $LD_LIBRARY_PATH  
[oracle@localhost sap]$ echo $SYBASE
```

Sample Output:

```
[oracle@localhost sap]$ echo $LD_LIBRARY_PATH
/opt/sap/OCS-16_0/lib:/opt/sap/OCS-16_0/lib3p64:
/opt/sap/OCS-16_0/lib3p:
[oracle@localhost sap]$ echo $SYBASE
/opt/sap/
```

Note

When creating the deployment, ensure that both the LD_LIBRARY_PATH and SYBASE variables are listed in the Environment Variables section of Step 3, of the Oracle GoldenGate Configuration Assistant. If not, manually add any variables that are missing, using the values from your system and continue with the deployment creation.

Prerequisites for Installing Oracle GoldenGate for Teradata

Learn about the prerequisites for installing Oracle GoldenGate for a Teradata database.

Install ODBC Drivers for Teradata

Creating an Oracle GoldenGate for Teradata deployment requires that a supported version of the Teradata ODBC driver be installed before launching the Oracle GoldenGate Configuration Assistant (OGGCA).

Install a supported Teradata ODBC driver based on the database version and operating system where Oracle GoldenGate will be installed. Use the following link to find the available Teradata ODBC drivers:

Use the following link to find the available ODBC drivers for Teradata:

<https://downloads.teradata.com/>

Review the README instructions provided by Teradata and complete the required driver installation steps.

Prerequisites for Installing Oracle GoldenGate for TimesTen

Learn the prerequisites for installing Oracle GoldenGate for a TimesTen database.

To create an Oracle GoldenGate for Oracle TimesTen deployment requires installing the Oracle TimesTen Client on the server where Oracle GoldenGate is to be installed. If Oracle GoldenGate is installed on the Oracle TimesTen database server, then the required client components are already available. However, if you are installing Oracle GoldenGate on a hub server, then you must separately install the Oracle TimesTen Client.

Oracle TimesTen Software Installation

Download the Oracle TimesTen software from the following site, choosing a version and operating system platform of the software based on the platform where Oracle GoldenGate is to be installed, and the version of the Oracle TimesTen database.

<https://www.oracle.com/database/technologies/timesten-downloads.html>

For a list of supported Oracle GoldenGate platforms and supported versions of the Oracle TimesTen database, review the [Certification Matrix](#) for this release of Oracle GoldenGate.

1. Download the In-Memory database for a supported version of the database.
2. Extract the Oracle TimesTen installation files to a designated location, based on the instructions provided in *Oracle TimesTen In-Memory Database Installation Guide*.
3. Follow the installation instructions provided by the TimesTen documentation for complete installation details.

Client-only Instance Creation

For non-database server environments where you plan to install Oracle GoldenGate, after installing the Oracle TimesTen client libraries, follow the TimesTen document instructions to create a client-only instance of TimesTen.

1. Perform the following:

```
[oracle@tt_installation_dir]$ ./bin/ttInstanceCreate -clientonly
```

2. Follow the instance installation prompts, taking note of the instance name and the instance installation location. This information will be required when setting up a Replicat's ODBC connection to Oracle TimesTen.

Setting the Environment Variables

Ensure that the required system environment variables are sourced before installing Oracle GoldenGate for TimesTen. The correct environment settings are needed for all sessions or processes that will interact with Oracle TimesTen. Every Oracle TimesTen instance (server and client) contains a script for setting the required environment variables. This script is located in the `tt_instance_home/bin` and is named `ttenv.[c]sh`. It should always be dotted or sourced and never executed directly.

Example of setting the bash shell environment for a TimesTen instance located in `/home/oracle/tt221`:

```
source /home/oracle/tt221/bin/ttenv.sh
```

Although it is possible to set the required environment variables manually, it is not recommended.

Prerequisites for Installing Oracle GoldenGate Remote Capture and Delivery

When installing Oracle GoldenGate on a remote server (one different from where the database is running), set the remote server's time and time zone to that of the source database server so that Oracle GoldenGate Extract can correctly position by time when creating the Extract with the `BEGIN` option, otherwise, position by a valid database log record value, such as an LSN or SCN.

Other Program and Settings

Learn about additional prerequisites for before installing Oracle GoldenGate Microservices Architecture for non-Oracle databases.

The following additional features 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 or any other zip file utility 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.
- For Oracle GoldenGate for Db2 z/OS, the objects require a minimum hardware platform of z10, a minimum operating system release 1.13, and a minimum Db2 release 11.

Oracle GoldenGate supports Sysplex data sharing.

3

Install, Patch, and Upgrade

Learn about the steps to install Oracle GoldenGate as well as post-installation tasks, installing patches, upgrading Oracle GoldenGate from a prior version, and uninstalling Oracle GoldenGate.

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. Refer to the following links, before you begin installing Microservices Architecture:

- [Operating System Requirements](#)
- [Prerequisites for Installing Oracle GoldenGate for Db2 LUW](#)
- [Prerequisites for Installing Oracle GoldenGate for Db2 z/OS](#)
- [Prerequisites for Installing Oracle GoldenGate for Db2 for i](#)
- [Prerequisites for Installing Oracle GoldenGate for MySQL](#)
- [Prerequisites for Installing Oracle GoldenGate for Oracle](#)
- [Prerequisites for Installing Oracle GoldenGate for SQL Server](#)
- [Prerequisites for Installing Oracle GoldenGate for Sybase](#)
- [Prerequisites for Installing Oracle GoldenGate for Teradata](#)
- [Prepare Databases](#)

To install Oracle GoldenGate Microservices Architecture (MA), perform the following steps:

1. Install the Oracle GoldenGate software using the steps given in [Performing an Interactive Installation with OUI for MA](#) or [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. Create client, server, and trusted rootCA Certificates for a secure GoldenGate deployment. The steps to create these certificates are available in [Create Certificates for Secure Deployments](#).
4. 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.

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.

Set Environment Variables for Oracle GoldenGate for Db2 z/OS

After installing Oracle GoldenGate for Db2 z/OS on the required operating system, you need to set the following environment variables, before running OGGCA for creating a deployment:

IBMCLIDRIVER: This needs to point to the Db2 clidriver directory.

LD_LIBRARY_PATH: This needs to point to the lib subdirectory under the clidriver directory.

If you do not set these environment variables for Db2 z/OS, the Oracle GoldenGate deployment creation will fail.

Patching Oracle GoldenGate

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 Oracle GoldenGate New Features for any new features, parameter changes, patching requirements, known issues, or bug fixes that affect your current configuration.

Patching 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_SRV_MGR_DIRECTORY/var
export OGG_ETC_HOME=OGG_SRV_MGR_DIRECTORY/etc
OGG_SRV_MGR_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
```

Post-Patch Installation Tasks for Non-Oracle Databases

This topic lists any post-patch installation tasks that may be required, depending on the database platform.

Post-Patching Tasks for Oracle GoldenGate for SQL Server

Update odbcinstant.ini and odbcinstant.ini

If patching Oracle GoldenGate 23.4 or 23.5, to version 23.6 or later, a one-time change to both the `odbcinst.ini` and `odbc.ini` is required to force connections to utilize the updated driver supplied with Oracle GoldenGate.

You do not need to follow these steps if patching from Oracle GoldenGate 23.6 or after, to later Oracle GoldenGate 23ai patch versions.

1. The `odbcinst.ini` file is located in the `$OGG_HOME/msodbc` directory. Modify this file to reflect the information provided below and ensure the ODBC Driver value includes the full path of the Oracle GoldenGate installation. For example:

```
[Microsoft ODBC Driver for SQL Server]
Description=Microsoft ODBC Driver for SQL Server
Driver=/OGG26ai_Install_Path/msodbc/lib64/libmsodbcsql.so
```

2. If you have previously created an `odbcinst.ini` file with DSN entries, update the ODBC Driver value to match that of the Description field in the `odbcinst.ini` file. It should be set to the following:

```
Driver=Microsoft ODBC Driver for SQL Server
```

Uninstalling Patches Using OPatch

To uninstall a patch for Oracle and Non-Oracle Databases, 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.

Upgrade

Learn about the tasks required for upgrading Oracle GoldenGate Microservices Architecture.

The Oracle GoldenGate upgrade process entails a major software version upgrade, from a lower version of the Oracle GoldenGate to the current version of the software, such as upgrading Oracle GoldenGate 21c to Oracle GoldenGate 26ai.

For Oracle GoldenGate 26ai, you can directly upgrade from 19c or 21c Microservices build to the 26ai build.

For Classic Architecture, you must first upgrade to the Classic Architecture builds for 19c or 21c release and then use the conversion utility to convert it to Oracle GoldenGate Microservices 26i. This is mandatory for earlier versions of Oracle GoldenGate such as 12.3 or 18.1.

Obtaining the Oracle GoldenGate Distribution

Refer to the instructions provided in the [Download Oracle GoldenGate Software](#) section to obtain the required upgrade build.

Prerequisites

Learn about prerequisites for upgrading Oracle GoldenGate Microservices Architecture.

As a best practice, perform a minimal or basic upgrade first, which implies performing the upgrade without adding any new features and additional or non-mandatory parameters.

If Oracle GoldenGate is upgraded at the source side where the Extract exists, then the trail file format remains the same. Only if a higher `FORMAT RELEASE` is adjusted to the `EXTTRAIL`

parameter or an ETR0LLOVER is performed, will the trail file get upgraded to a higher release. This provides the opportunity to upgrade the target system where the Replicat exists, independently. When all target systems are upgraded, you may update the format release of the EXTTRAIL parameter to leverage new features that rely on a higher trail file format. No repositioning of any process is required.

After you verify that the environment is upgraded successfully, you can implement the new features and additional parameters as required.

The upgrade instructions also include the steps for upgrading the source or target database and Oracle GoldenGate at the same time.

Oracle GoldenGate Upgrade Considerations

Before you start the upgrade, review the information about upgrading Extract and Replicat.

As a best practice, perform a minimal or basic upgrade first, which implies performing the upgrade without adding any new features and additional or non-mandatory parameters. After you verify that the environment is upgraded successfully, you can implement the new features and additional parameters as required.

Even though you may only be upgrading the source or target installations, 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.

Installation Binaries and Deployments

With Microservices Architecture, there is a strong separation between where the software is installed and the deployment directory structure for the Oracle GoldenGate instance, which contains the parameter files, report files, and trail files. For both these areas, the software binaries and deployment, are strictly separated. So, there is no interference between the old and new software installations related to the deployments.

During a software upgrade, the new software will be installed independently. The deployment working with the old software will be stopped. Then, the deployment environment will be adjusted to the new software and the deployment will be restarted.

If you have a reverse proxy configuration on your host machine generated with `OGG_HOME/lib/utl/reverseproxy/ReverseProxySettings`, then consider reconfiguring it to leverage the enhanced `ReverseProxySetting` utility available with Oracle GoldenGate 21c (21.3) and higher releases. For details on `ReverseProxySetting` utility, see [Configure Oracle GoldenGate Reverse Proxy with NGINX](#).

Security Considerations

You cannot change an unsecure deployment to secure deployment or a secure deployment to an unsecure one. Deployment Security must be configured at Deployment creation time, and cannot be changed. If you want to change to a secure deployment, you need to create another secure deployment and then manually move Extracts, Replicats, Receiver Service, and Distribution Service to that new deployment.

Service Manager and Deployments

When upgrading Oracle GoldenGate, the Service Manager must be updated first. The software version of the Service Manager must be higher or equal to the version of the deployments, except when XAG is enabled.

Note

If using Oracle Grid Infrastructure Agent (XAG), the deployments managed by the upgraded Service Manager must also be upgraded. It is not supported to run any deployments under a Service Manager using a different Oracle GoldenGate software release than that of the Service Manager when XAG is enabled.

Extract Upgrade Considerations

If you are upgrading multiple Extract processes that operate in a consolidated configuration (many sources to one target), upgrade one Extract at a time.

The output trail file is automatically rolled over when the Extract restarts and the Extract version is upgraded.

For Oracle GoldenGate for the Oracle Database, when a database is upgraded to a higher binary version and then downgraded to a lower binary version, any existing captures cannot continue mining. The captures have to be dropped and new Extracts have to be created in order to continue.

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.

When upgrading from releases prior to 19c release of Oracle GoldenGate, 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.

The `TIMEZONE` datatype is managed differently with Oracle GoldenGate 21c and higher releases. Therefore, you may need to run the `ALTER REPLICAT EXTSEQN0` command to synchronize with newer trail files after consuming the old trail file written by the Extract. See the `ALTER REPLICAT` command for details.

Trail File Considerations

If Oracle GoldenGate is upgraded at the source side where the Extract exists, then the trail file format remains the same. However, the trail file gets upgraded to a higher release only if a higher `FORMAT RELEASE` is adjusted to the `EXTTRAIL` parameter or an `ETROLLOVER` is performed. This provides the opportunity to upgrade the target system where the Replicat exists, independently. When all target systems are upgraded, you may update the format release of the `EXTTRAIL` parameter to leverage new features that rely on a higher trail file format. No repositioning of any process is required.

Pre-upgrade Steps

Learn about prerequisites for upgrading to Oracle GoldenGate Microservices Architecture.

- Install the most recent version of the Oracle GoldenGate release to which you want to upgrade. For installation steps, see [Installing Oracle GoldenGate Microservices Architecture](#).
- If the Service Manager to be upgraded is managed using the Oracle Grid Infrastructure Agent (XAG), you must disable the Oracle GoldenGate instance before the upgrade to prevent the regular XAG healthchecks from failing during the upgrade process:

```
$ agctl disable goldengate instance_name
```

Upgrade Oracle GoldenGate Microservices – GUI Based

Learn the steps to upgrade Oracle GoldenGate Microservices using the GUI.

At this point, you should have a new Oracle GoldenGate 26ai Microservices home and any prior release homes of Oracle GoldenGate MA.

Upgrade the Service Manager

After installing the latest Oracle GoldenGate MA version, the next step is to upgrade the Service Manager:

1. Log into Service Manager from the URL: *https://hostname:servicemanager_port*
2. Expand the **ServiceManager** link in the **Deployments** section of the Service Manager Overview page.
3. Click **Details** and then the pencil icon next to the Deployment Detail section to open the dialog box for editing the **GoldenGate** home path.
4. Update the Oracle GoldenGate Home path with the full path to the new Oracle GoldenGate home.
5. Click **Submit**.
6. Return to the **Deployments** section and click the **Restart** icon in the **Actions Field** to restart the Service Manager.

Upgrade a Deployment

To upgrade a deployment:

1. Stop all Extract and Replicat processes gracefully:
 - Check for open (long running) transaction and Bounded Recovery as it may take longer to stop Extract gracefully.
 - If any unnecessary open transactions are visible, for example `SEND EXTRACT group_name SHOWTRANS`, then those transactions can be skipped or immediately forced to stop. In this case, a Bounded Recovery checkpoint can be retrieved using the following command:

```
SEND EXTRACT group_name, BR BRCHECKPOINT immediate
```
2. Verify the current location of Oracle GoldenGate 26ai home directory from the Service Manager.
 - a. Login to the Service Manager: *http://hostname:servicemanager_port*
 - b. Expand the ServiceManager link in the Deployments section of the Service Manager Home page.

- c. Copy the **Oracle GoldenGate Home** path.
3. Return to the **Deployments** section and expand the link of the deployment to be upgraded.
4. Click **Details** and then the **pencil** icon next to the **Deployment Detail** section to open the dialog box for editing the **GoldenGate Home** path.
5. Update the Oracle GoldenGate Home path with the complete path to the new Oracle GoldenGate home directory.
6. Click **Submit** and confirm that the Oracle GoldenGate Home path has been updated.
7. Depending on the database of the deployment being upgraded, there may be additional changes to the Environment Variable settings. Click **Configuration** to modify the settings for the environment variables, if required.

(Oracle GoldenGate for Oracle Database only), The Unified Build feature requires only the TNS_ADMIN to be adjusted to the Oracle Database Client software within Oracle GoldenGate.

TNS_ADMIN = *Location of tnsnames.ora and sqlnet.ora*

The OGG_HOME environment variable can automatically detect the DATABASE_HOME, LD_LIBRARY_PATH, and JAVA_HOME.

Note

When upgrading from 21c to 26ai, this step is not necessary. If upgrading from Oracle GoldenGate 19c (or earlier) to 26ai, you only need the TNS_ADMIN environment variable.

(Oracle GoldenGate for SQL Server only) Oracle GoldenGate 26ai for SQL Server on Linux includes an ODBC driver for database connectivity. Although it's not required, it is recommended to utilize the packaged driver. To use the packaged driver, two new environment variables need to be created and any previously created `odbc.ini` file needs to be updated.

Add the following environment variables to the deployment:

- OBCSYSINI = `${OGG_HOME}/msodbc`
- ODBCINI = `${OGG_HOME}/msodbc/odbc.ini`

After creating the new environment variables, copy your existing `odbc.ini` file to the GoldenGate `msodbc` folder and change the ODBC Driver value of any DSN entry to the following:

Driver=Microsoft ODBC Driver for SQL Server

8. Click **Submit** to save the changes made to the Environment Variables.
9. Return to the Service Manager Home page.
10. Click the **Restart** icon in the **Actions** field for the deployment and click **OK** to restart the deployment.
11. Log back into the new Administration Service and start all Extracts and Replicats.

Note

For Oracle GoldenGate for PostgreSQL, see [Post-Upgrade Steps](#) prior to starting any Extracts.

When the Service Manager and deployments have been restarted, the upgrade is complete.

Upgrade Oracle GoldenGate Microservices Using REST APIs

Learn the steps to upgrade Oracle GoldenGate Microservices using the REST APIs.

At this point, you should have a new Oracle GoldenGate 26ai MA home and any prior release homes of Oracle GoldenGate MA.

Upgrade the Service Manager

After installing the latest Oracle GoldenGate MA version, the next step is to upgrade the Service Manager.

When upgrading the Service Manager, you can use the following cURL example to update the Oracle GoldenGate home:

```
curl -u adminname:adminpwd -X PATCH
\ https://hostname:port/services/v2/deployments/ServiceManager
\ -H 'cache-control: no-cache'
\ -d '{"oggHome":"new OGG_HOME_absolute_path", "status":"restart"}'
```

In this syntax, enter the new Oracle GoldenGate home directory absolute directory path such as /u01/app/oracle/product/26ai/gghome_1.

If not using XAG, check if Service Manager is running from the new \$OGG_HOME, using the following command:

```
ps -ef | grep -i servicemanager
```

If you don't see Service Manager in running state, then run the following command:

```
cd $NEW_OGG_HOME/bin
$ ./ServiceManager
```

Upgrade a Deployment

Deployments can be upgraded in the same step with the Service Manager, or they can be upgraded at a later time after the Service Manager has been upgraded.

To upgrade a deployment:

1. Stop all Extract and Replicat processes gracefully:
 - Check for open (long running) transaction and Bounded Recovery as it may take longer to stop Extract gracefully.
 - If any unnecessary open transactions are visible, for example `SEND EXTRACT group_name SHOWTRANS`, then those transactions can be skipped or immediately forced to stop. In this case, a Bounded Recovery checkpoint can be retrieved using the following command:

SEND EXTRACT *group_name*, BR BRCHECKPOINT immediate

2. (Oracle GoldenGate for Oracle Database only) The Unified Build feature requires only the TNS_ADMIN to be adjusted to the Oracle Database Client software within Oracle GoldenGate.

```
curl -k -u adminname:adminpwd -X PATCH
\ https://server.oracle.com:9000/services/v2/deployments/uat_01
\ -H 'cache-control: no-cache'
\ -d '{"environment": [ , {"name": "TNS_ADMIN" , "value": "/u01/app/oracle/
network/admin"} ] }'
```

Note

When upgrading from 21c to 26ai, this step is not necessary. If upgrading from Oracle GoldenGate 19c (or earlier) to 26ai, you only need the TNS_ADMIN environment variable.

The OGG_HOME environment variable can automatically detect the DATABASE_HOME, LD_LIBRARY_PATH, and JAVA_HOME.

3. (Oracle GoldenGate for SQL Server only) Oracle GoldenGate 26ai for SQL Server on Linux includes an ODBC driver for database connectivity. Although it's not required, it is recommended to utilize the packaged driver. To use the packaged driver, two new environment variables need to be created and any previously created `odbc.ini` file needs to be updated.

Add the following environment variables to the deployment:

```
curl -k -u adminname:adminpwd -X PATCH
\ https://server.oracle.com:9000/services/v2/deployments/uat_01
\ -H 'cache-control: no-cache'
\ -d '{"environment": [ , {"name": "ODBCSYSINI" , "value": "${OGG_HOME}/
msodbc"}
, {"name": "ODBCINI" , "value": "${OGG_HOME}/
msodbc/odbc.ini"} ] }'
```

After creating the new environment variables, copy your existing `odbc.ini` file to the Oracle GoldenGate `msodbc` folder and change the ODBC Driver value of any DSN entry to the following:

Driver=Microsoft ODBC Driver for SQL Server

4. Run this cURL command to upgrade the Oracle GoldenGate deployment:

```
curl -u SM username:SM password -X PATCH
http://hostname:servicemanager port/services/v2/deployments/Deployment-
name
-H 'cache-control: no-cache'
-d '{"oggHome":"new OGG_HOME complete path","status":"restart"}'
```

5. Start all Extracts and Replicats.

Note

For Oracle GoldenGate for PostgreSQL, see [Post-Upgrade Steps](#) prior to starting any Extracts.

When the Service Manager and deployment have been restarted, the upgrade is complete.

Post-Upgrade Steps

Learn about the post-upgrade tasks for different upgrade scenarios.

There are multiple post-upgrade steps that you must complete before beginning Oracle GoldenGate data replication set up. These tasks are:

- Upgrading the XAG Configuration
- Upgrading the heartbeat table
- (Oracle GoldenGate for PostgreSQL only) Upgrading the PostgreSQL Extract.

Update the XAG Oracle GoldenGate Instance Configuration

If XAG is in use, and disabled prior to the upgrade process, you must update and re-enable the GoldenGate instance after upgrading the Service Manager and deployments.

1. Update the GoldenGate instance parameters with the new Oracle GoldenGate home and Oracle Home directories:

```
$ agctl modify goldengate instance_name
--gg_home New OGG_HOME directory
--oracle_home New OGG_HOME directory/lib/instantclient
```

Note

Since the introduction of the Oracle GoldenGate 21c software unified build, the ORACLE_HOME directory is contained in \$OGG_HOME/lib/instantclient.

2. Re-enable the Oracle GoldenGate instance:

```
$ agctl enable goldengate instance_name
```

Upgrade the Heartbeat Table

After completing the upgrade, run the `UPGRADE HEARTBEATTABLE` command against any source and target databases where it was previously created, to add extra columns for tables and lag views. These extra columns are used to track the Extract restart position. See `UPGRADE HEARTBEATTABLE` to know more.

Upgrade Extract for Oracle GoldenGate for PostgreSQL

The Oracle GoldenGate CDC Capture support from PostgreSQL relies on using replication slots on the PostgreSQL database to read the change records from WAL. The replication slot name, used by each CDC Extract, depends on the Oracle GoldenGate installation path.

You may want to use the same Extract from the exiting Oracle GoldenGate environment to continue capturing the change records in the new Oracle GoldenGate environment, after the

upgrade although Extract has not finished capturing the change records till the time of upgrade. However, this will not be possible if a new Oracle GoldenGate installation directory is used for the upgrade. You must use the existing Oracle GoldenGate installation directory to make this work.

To upgrade the older Oracle GoldenGate version or migrate from Oracle GoldenGate Classic Architecture to Microservices Architecture, use the following steps:

Note

The following steps use an Extract name `extn` as an example.

1. Obtain the replication slot name that was being used by the Oracle GoldenGate CDC Extract in the older environment. The old replication slot name can be obtained using from the Extract report file or by running the `INFO EXTRACT` command.

From Extract report file in the older Oracle GoldenGate environment. Following is the example snippet from the Extract report file:

```
2024-01-09 11:49:38 INFO    OGG-25376 Oracle GoldenGate capture 'EXTN'
running with replication slot 'extn_5d87a7db5e810943', slot type
'logical', plugin name
as 'test_decoding' attached with database 'postgres' with slot restart LSN
as '6/3CBFE2B8', flush LSN as '6/3CBFE2F0' and its current TXID as
'7899713'.
```

Note the replication slot name from the Extract report file. In this example it is `extn_5d87a7db5e810943`.

You can also run the `INFO EXTRACT` command from GGSCI or Admin Client in the older Oracle GoldenGate environment, as shown in the following example:

```
GGSCI (phoenix96567 as postgres@pg12) 10> info extract extn
```

Output:

```
Extract      EXT1      Last Started 2024-01-09 11:49  Status RUNNING
Checkpoint Lag      00:00:45 (updated 00:00:01 ago)
Process ID          3640439
VAM Read Checkpoint 2024-01-09 11:48:52.575833
Replication Slot     extn_5d87a7db5e810943 is active with PID 3640450 in
database postgres
Slot Restart LSN     6/3CBFE2B8
Slot Flush LSN       6/3CBFE2F0
Current Log Position 6/3CBFE2F0
```

2. Obtain the replication slot name that is relevant for the Oracle GoldenGate CDC Extract in the newer Oracle GoldenGate environment. Run the `UNREGISTER EXTRACT` command to obtain the new replication slot name:

```
unregister extract extn
```

Output:

```
2024-01-09 11:43:37 INFO    OGG-25354 The replication slot
'extn_6ec0db7cc27c812' for group 'EXTN' does not exist in database
'postgres'.
```

Note the replication slot name from the error message. In this example it is `extn_6ec0db7cc27c812`.

3. Run the following SQL query on the corresponding source PostgreSQL database (PSQL can be used):

```
SELECT * FROM pg_copy_logical_replication_slot('old-replication-slot',
'new-replication-slot');
```

For example:

```
SELECT * FROM pg_copy_logical_replication_slot('extn_5d87a7db5e810943',
'extn_6ec0db7cc27c812');
```

4. Make sure to copy the Extract checkpoint files, trail files, and other configuration files from old Oracle GoldenGate environment to the new environment.
5. Check and confirm that the LSN positions of *new-replication-slot* is same as the *old-replication-slot*.
6. Run the `UNREGISTER EXTRACT extract_name` command in the old Oracle GoldenGate environment. For example:

```
UNREGISTER EXTRACT extn
```

7. Start the Extract in the new Oracle GoldenGate environment to resume capturing of change records.

Migrate from Classic to Microservices Architecture Using the Migration Utility

The Oracle GoldenGate Classic to Microservices Migration Utility is used to migrate from an Oracle GoldenGate Classic deployment to a Microservices deployment. This utility is an interactive utility that migrates Extracts, Replicats from a Classic deployment to the Microservices deployment and converts pumps to a distribution path. It also migrates trail files, checkpoints, and other associated artifacts.

The Migration Utility is available in the My Oracle Support (MOS) portal as Patch 37274898. Updates to the patch are released when required. The patch description includes details with the latest build number.

Prerequisites for Using the Migration Utility

Following are the prerequisites for using the Migration Utility:

- Oracle GoldenGate 19c and higher with Bundle Patch 200714 or later.

Note

The Migration Utility can also be used with Oracle GoldenGate 21c, if the underlying database/system already supports Oracle GoldenGate Microservices Architecture in the earlier version, such as Oracle GoldenGate for DAA. If the underlying database does not support Microservices Architecture, then you can use Oracle GoldenGate 23ai for heterogeneous databases.

- Java version 1.8 JRE or later (Same as provided by Oracle GoldenGate).
- New deployments created using Microservices Architecture should be local to the Classic Architecture deployments or the Classic Architecture home file system must be available on the same host.
- New Microservices Architecture deployment created and all services running.
- Classic Architecture deployment must be stopped during migration. Make sure that the server collectors (if any) are also stopped. Abended processes must be restarted and gracefully shutdown.
- The Migration Utility needs to run from a shell/command prompt that has the environmental variables set for the Microservices Architecture deployment.

The environment variables for OGG_HOME must be set to the Oracle GoldenGate Microservices software directory location.

On Microsoft Windows, the PATH variable must point to OGG_HOME\lib.

On UNIX and Linux platforms, the LD_LIBRARY_PATH needs to point to OGG_HOME/lib.

Migration Utility Restrictions

The migration utility cannot migrate any processes that have the following

- Extract Pump processes that perform transformations or has a DBLOGIN.
- Classic Extract (Deprecated in Oracle GoldenGate 19c and desupported in version 21c for any target MA greater than 19.1.)
- Processes that use the following parameters:
 - MACRO
 - OBEY
 - MACROCHAR
- Use of Remote Trail Files, Remote Tasks or Initial Load.
- Absolute path names for trail files. Classic trail file MUST reside in ./dirdat (or a subdirectory within it). If the trail file reside in another directory, they can be moved to ./dirdat and then modify the checkpoint using the convchk utility for the Extract. Then modify the parameter for any Replicat that is affected. See convchk in the *Parameters and Functions Reference for Oracle GoldenGate* for Oracle GoldenGate 21c.
- INCLUDE Restrictions:
 - Nested INCLUDES
 - Only in Extract and Replicats, not in Extract Pump.
- Oracle GoldenGate for DAA (Only): For each Extract and Replicat group, one should manually inspect the corresponding .properties file for properties whose value refers to a

file in the Classic file system. For each of these properties, do the following manual migration steps:

1. Copy the referenced file to the Microservices deployment file system.
2. Modify the property value to refer to the new file path in the Microservices deployment file system.

Run the Migration Utility

Make sure that you have completed the steps mentioned in [Prerequisites for Using the Migration Utility](#) before you begin the tasks in this section.

Use the following steps on the command prompt to run the Migration Utility:

1. Run the following command to start the Migration Utility:

```
java -jar ogg-migration-package-23.4.0.0.0.jar -classicHome /scratch/ogg/home -
```

```
-dryrun java -jar ogg-migration-package-23.4.0.0.0.jar -classicHome /scratch/ogg/home
```

```
-debug java -jar ogg-migration-package-23.4.0.0.0.jar -help
```

The options available with this command are:

- `-classicHome path`: (Required) Must point to the location of the Oracle GoldenGate Classic software directory.
- `-dryrun`: (Optional) Indicates that the migration is simulated (Do a Dry Run) to check for possible issues.
- `-debug`: This is optional but it is recommended that you use this option with the utility. Enables debug level logging and diagnostics. It provides more detailed log data and will produce a zip file with artifacts from both the classic and MA when a migration fails.
- `-skipPumpURI`: Specifies the **Accept Default URI on Pump Migrate**
- `-dbtype`: Specifies the database name. Valid values are 'ORACLE', 'MYSQL', 'MSSQL', 'POSTGRESQL', 'SYBASE', 'DB2400', 'DB2ZOS', 'DB2LUW', 'TERADATA', 'TIMESTEN'
- `-help`: Shows usage.

2. The utility prompts for the user credentials for the Service Manager of the MA deployment:

```
*****
**      Oracle GoldenGate Classic to Microservices Migration
Utility      **
**      Version 23.4.0.0.0 Build 66 at Fri Jan 22 17:24:41 UTC
2024      **
**      Copyright (C) 1995, 2024, Oracle and/or its affiliates. All rights
reserved.  **
**
**
**      Operating system character set identified as
```



```

UTF-8.                **
*****
*****

```

```

Command Entered: -classicHome /ogg/payroll -debug
Log File for this session is /ogg/payroll/./Migration081321105419.log.
Debug Mode is Enabled.
Enter Service Manager Credentials: User Id: dba
Password:

```

3. The utility scans for a Service Manager on the host and provides a choice of deployments to migrate to if there are more than one deployment and then prompts to continue.

```

Scanning for Microservices Deployments...
Using Deployment payroll [cf4c7c93-3a1c-4cbc-b1ab-bc942bab7645]
The following processes will be migrated to deployment payroll:
Extracts: EXTE
Replicats: REPW
Pumps: DPE
Do you wish to continue? (yes/no):>

```

Enter yes to continue.

4. The utility will migrate the database credentials from the Classic deployment to the target Microservices deployment and then copy all the trail files in the Classic deployment and migrate all the Extracts and Replicats processes.

```

Migrating Credentials from /ogg/payroll/./dircrd
Migrating Credentials for Domain Payroll with an alias of ogg.
Migrating Credentials for Domain Payroll with an alias of user.

```

```

Copying Trail Files from /ogg/payroll/dirdat to /ogg/Microservices/
payroll/var/lib/data.
235 trail files copied.

```

```

Migrating EXTRACT EXTE to http://10.20.0.221:9901.
Checkpoint File(s) Copied and Converted Successfully.
Parameter File EXTE.prm Saved Successfully.
Migration of Extract EXTE Completed Successfully.

```

```

Migrating REPLICAT REPW to http://10.20.0.221:9901.
Parameter File REPW.prm Saved Successfully.
Checkpoint File(s) Copied and Converted Successfully.
Migration of Replicat REPW Completed Successfully.

```

5. The utility then migrates all the Extract Pump processes and prompt for the URL for the target Receiver Service for each. To accept the default value, press enter.

If you are using a different target, enter the URL. By default, the utility assumes that the classic pump is sending to another classic deployment server process. The default URL allows the Distribution Service to link to the classic target. If you are also planning to migrate the target Classic deployment at a later time, you will need to modify the newly created Distribution Path at the same time. See [Different Use Cases for Migration](#) for details.

Note

The loopback pumps are common in a test environment where a Classic pump sends to the same Manager on the same deployment. This loopback pump is automatically be routed to the Receiver Service on the migrated MA deployment.

Migrating PUMP PMPW2E to http://10.20.0.221:9902.

Use Default Pump Target URI of ogg://ushost1:9903/services/v2/targets?trail=../bb?

Press Enter to Accept or type Alternate URI:>

Creating Distribution Path for Pump PMPW2E.

Distribution Path Created Successfully.

Setting trail sequence to 235 with an RBA of 1319472691.

Migration of Pump PMPW2E Completed Successfully.

Migrating Manager tasks to http://10.20.0.221:9901.

Creating Task for Parameter PurgeTrail-aa. Task Created Successfully.

Creating Task for Parameter Purge-Extract. Task Created Successfully.

2 Tasks were migrated, 2 Successfully, 0 with errors.

Migrating Lag Report.

Creating Task for Parameter LagReport. Task Created Successfully.

Migration pf Manager Tasks Completed Successfully.

6. Next, the utility asks if it should migrate parameters in the GLOBALS configuration file and report on any issues encountered.

Do you wish to migrate your GLOBALS parameters? (yes/no):> y

GLOBAL parameters successfully migrated, 3 parameters merged.

7. The utility prompts to copy existing wallet entries if they exist and issues a warning if entries exist on the target wallet. These target wallet entries will be overwritten. After this task completes, the utility terminates.

Your Classic Deployment has existing wallet entries.

Do you wish to copy them? (yes/no):> y

Copying Oracle Wallet Entries.

WARNING: Target deployment has existing wallet entries and will be overwritten!

Do you wish to continue? (yes/no):> y

Wallet entries successfully copied.

Migration Processing Complete, 3 processes were migrated successfully.

If the migration was successful, it will rename the checkpoint files in the Classic dirchk directory by appending .migrated to the name. This will prevent the Classic deployment processes from being started inadvertently.

If the utility encounters an issue and is not able to complete the migration process, then it will perform a rollback of all of the work it has performed and restore the Classic deployment to its original state. It will produce a diagnostic zip file containing the log file, classic configuration and target configuration to help diagnose the issue.

Different Use Cases for Migration

Oracle GoldenGate Classic Architecture deployments are usually not self-contained. Deployments typically send trail data to other deployments using the Pump process that have very complex interconnected topologies. When a deployment is migrated to Microservices, it still may be sending trail data to another Classic or Microservices deployment. A major feature of Microservices Architecture is the ability to secure the deployment using established security protocols such as TLS/SSL and secured web sockets.

When migrating deployments, a well thought out approach should be used. Usually, it is preferred that source and target deployments are migrated on an individual basis and not all at once. The Migration Utility allows you to migrate deployments one by one. As the source deployment uses a non-secured protocol to communicate with Classic Server/Collector or the Receiver Service, the migrated Distribution Path will use the legacy **ogg** protocol in the path. While this will be functional, it does not take advantage of the security capabilities of Microservices Architecture in creating a fully secure deployment.

It is recommended that you replace the Distribution Path generated by the Migration Utility, with one using the **wss** or **ws** protocols after all deployments have been migrated to Microservices Architecture or whenever the source and target deployments have been migrated.

To allow phased migration of deployments and processes to Microservices Architecture, source and target Classic Architecture deployments can be independently migrated.

The following lists different use cases and the associated Distribution protocol used while setting up the distribution path.

Source	Target	Distribution Protocol	Description
Hub architecture (no distribution trails) Migrate Classic to Microservices	Not Applicable	Not Applicable	Not Applicable
Migrate Classic to Microservices	Classic Deployment	ogg protocol to target classic deployment	-
Migrate Classic to Microservices	Microservices Deployment	ogg protocol	Optionally, drop and recreate DISTPATH to use ws or wss protocol (if target is secured with a reverse proxy)
Classic deployment	Migrate Classic to Microservices	Pump to target Receiver Service	Source Pump must be modified: Receiver Service Port and Remote Trail
Microservices deployment	Migrate to Classic to Microservices	ogg protocol	Drop and recreate DISTPATH and use ws or wss protocol (if target is secured with a reverse proxy)

Migrating Source from Classic to Microservices with Target as Microservices

In this use case, you can drop and recreate the Distribution Path to use the **ws** or **wss** protocol (if target is secured with a reverse proxy) by specifying the sequence no (seqno) and RBA values. This is optional as the target deployment is using Microservices Architecture.

Run the Admin Client and connect to the source deployment to get the SEQNO and RBA values:

```
OGG (http://north:9001)> INFO DISTPATH dpe, DETAIL
Path Name: dpe Status: running
Source: trail://north:9002/services/v2/sources?trail=ea
Target: ogg://south:9003/services/v2/targets?trail=ra
Authentication Method: USERIDALIAS ggsca domain OracleGoldenGate
[...]
Checkpoints:
Input Checkpoint:
Current:
  Name: ea
  Sequence: 14
  Offset: 2579
  Path: /u01/ogg_deployments/depl_north/var/lib/data/
[...]
```

Convert the distribution path to **ws** or **wss** protocol by recreating and repositioning the Distribution path that was created by the Migration Utility:

```
OGG (http://south:9001)> DELETE DISTPATH dpe, DETAIL

##If using Secure Web Socket Protocol (wss)
OGG (http://south:9001)> ADD DISTPATH dpm
                        SOURCE trail://src_host:9002/services/v2/sources?
trail=ea
                        TARGET wss://trg_host:9003/services/v2/targets?
trail=ra
                        AUTHENTICATION CERTIFICATE default
##If using Web Socket Protocol (ws)
OGG (http://south:9001)> ADD DISTPATH dpm
                        SOURCE trail://src_host:9002/services/v2/sources?
trail=ea
                        TARGET ws://trg_host:9003/services/v2/targets?
trail=ra
                        AUTHENTICATION USERIDALIAS ggsca DOMAIN
OracleGoldenGate
OGG (http://south:9001)> ALTER DISTPATH dpm, BEGIN SEQNO 14, RBA 2579
OGG (http://south:9001)> START DISTPATH dpm
```

Migrate Target from Classic to Microservices where Source is Microservices

In this use case, it is mandatory to drop and recreate the Distribution Path to use **ws** or **wss** protocol, if the target is secured with a reverse proxy. Follow the steps in [Migrating Source from Classic to Microservices with Target as Microservices](#) to drop and recreate the Distribution Path on the target deployment.

Migrating Target from Classic to Microservices with Source as Classic

When migrating the target deployment to Microservices Architecture while the source deployment is on Classic Architecture, you must perform the following additional steps:

- Modify the Pump port at source routing it to the target Receiver Service.
- The dirdat directory is not needed in the target Microservices file hierarchy. Modify RMTTRAIL file from ./dirdat/ra to ./ra in Extract Pump parameter file, as shown in the following example:

As shown in the following example, the target manager port number has been changed to the Receiver Service port number **9903** and the RMTTRAIL file path has been modified in the Extract Pump parameter file.

```
GGSCI> STOP EXTRACT PMPW2E
GGSCI> EDIT PARAMS PMPW2E
```

The following is the Extract Pump parameter file:

```
EXTRACT PMPW2E
USERIDALIAS ggeast,
target uri: host1:9903/services/v2/targets?trail=../bb? ENCRYPT AES192,
RMTTRAIL ../rt SEQUENCE hr.employees_seq; TABLE hr.*;
```

Restart Extract Pump after completing the modifications:

```
GGSCI> START EXTRACT PMPW2E
```

Note

No trail files are delivered to the target.

Before the migration begins, stop all processes at the source gracefully.

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	• /etc/init.d/OracleGoldenGate • /etc/rc.d/*OracleGoldenGate • /etc/rc*.d/*OracleGoldenGate • /etc/oggInst.loc
Linux 7 and Linux 8	/etc/systemd/system/ OracleGoldenGate.service

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.

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

```

Uninstalling Microservices Architecture Using Silent Mode

Note

It's important to remove all deployments prior to uninstalling Oracle GoldenGate home directory.
See [Files to be Removed Manually](#) for steps that you may need to perform manually.

To uninstall Oracle GoldenGate Microservices Architecture with Oracle Universal Installer silent mode:

1. Navigate to the following directory:

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

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

2. Run the command:

```
deinstall.sh -silent
```


Here's the output:

```
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_x21300x/local/ggtest/
install_202214
      (yes/no)? [no] yes
Starting Oracle Universal Installer...
Checking swap space: must be greater than 500 MB.
Actual 11648 MB
Passed
Preparing to launch Oracle Universal Installer from /tmp/
OraInstall2022-08-19_10-52-30AM.
      Please wait ...
Oracle Universal Installer, Version 21.1.3.0 Production Copyright (C)
1999, 2022, Oracle. All rights reserved.Starting deinstall
Deinstall in progress (Wednesday, August 19, 2022 10:52:33 AM
PDT)..... 100%
Done.
Deinstall successful
```

4

Secure

Learn about how to protect and secure your Oracle GoldenGate environment, including details on TLS support, authentication and authorization, trail file encryption options, streaming protocols, and other features that form the Oracle GoldenGate security framework.

Oracle GoldenGate security documentation is divided into Oracle GoldenGate Security Features and Oracle GoldenGate Security Features: Implementation.

[Oracle GoldenGate Security Features](#) section describes the of security standards applied with Oracle GoldenGate. This section is aimed to help Chief Security Officers (CSOs) learn about Oracle GoldenGate security features and standards.

[Oracle GoldenGate Security Feature: Implementation](#) section contains steps to implement various advanced security features available with Oracle GoldenGate. This section is aimed to assist DBAs and Oracle GoldenGate administrators implement security features available within Oracle GoldenGate.

Oracle GoldenGate Security Standards

This section provides details about the Oracle GoldenGate Security standards followed with the current release.

To view a list of Oracle GoldenGate security features, see [Oracle GoldenGate Security Features](#).

Secure Deployments

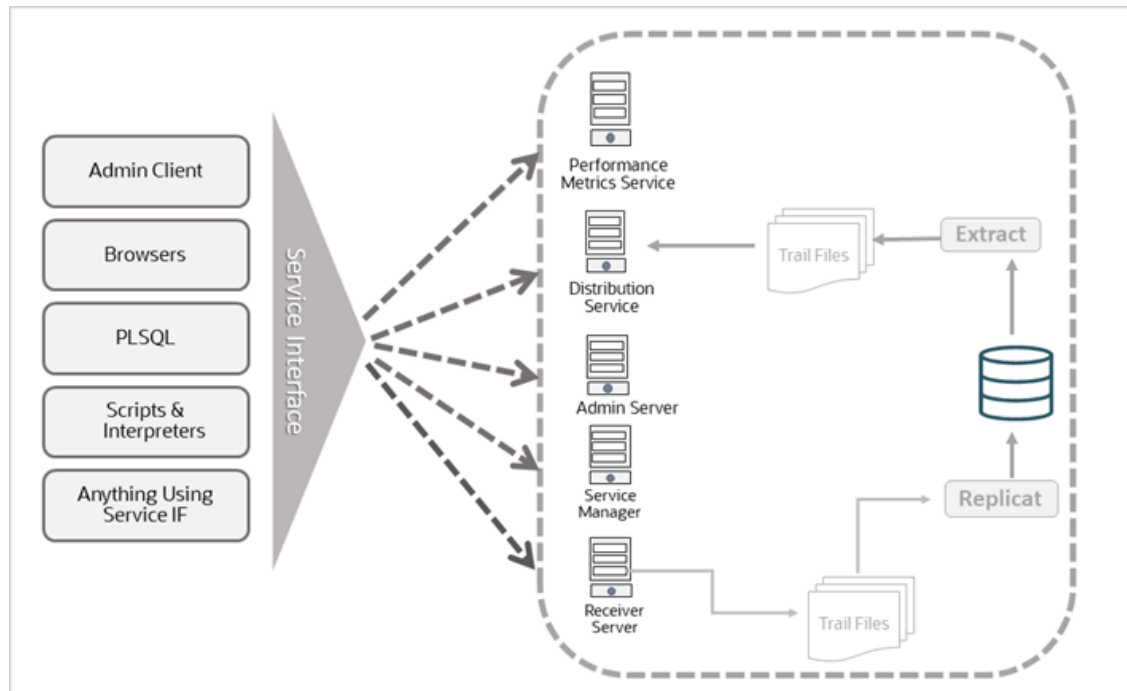
Oracle GoldenGate deployment can be secured in two ways:

1. Secure an Oracle GoldenGate deployment with Server Certificates. To set up server, client, and trusted root certificates from the Oracle GoldenGate web interface, see [Manage Certificates for Deployments](#). To use REST API service endpoints for setting up certificates for a deployment, see [Certificates REST Endpoints](#) in the REST API documentation.
2. Set up an Oracle GoldenGate deployment without a Server Certificate, but using a reverse proxy. TLS is terminated at the proxy. The server must be locked-down. To configure reverse proxy using Nginx, see [Configure Oracle GoldenGate Reverse Proxy with NGINX](#).

Note

mTLS is not supported when a Reverse Proxy is used.

This section discusses how to ensure that the Oracle GoldenGate deployment is secure. Using TLS for RESTful APIs ensures that all communication between the API consumers and the API endpoints is secure.



To learn more about these different RESTful interfaces available for accessing Oracle GoldenGate Microservices:

- See [REST API](#).
- See [About Admin Client](#) for commands to connect and use the deployment and Oracle GoldenGate processes.

You can access a deployment from any client machine using these interfaces.

Authentication and Authorization

Learn about the authentication and authorization features on the Oracle GoldenGate side and on the Database side when working with Oracle GoldenGate.

Oracle GoldenGate Authentication and Authorization

In Oracle GoldenGate Microservices, security is applied at both the authentication and authorization layers. Authentication can be managed using the following options:

- Using credentials (username, passwords): The credentials information is stored in a credential store, which is used to set up USERIDALIAS while setting up database connections. This is the most common method for authentication. Credential store can only be modified from REST API, Admin Client or Web Interface only.
- Using Certificates: If a client is authenticated by a certificate, the server stores the signing CA certificate in a trust store so that the server can validate the identity of the client. This is commonly used when using Distribution Paths.
- Using external Identify Providers: Oracle uses IDCS, IAM (Cloud) , and OAM (On-premise) for external Identity Management, using OAuth2 as the communication protocol.

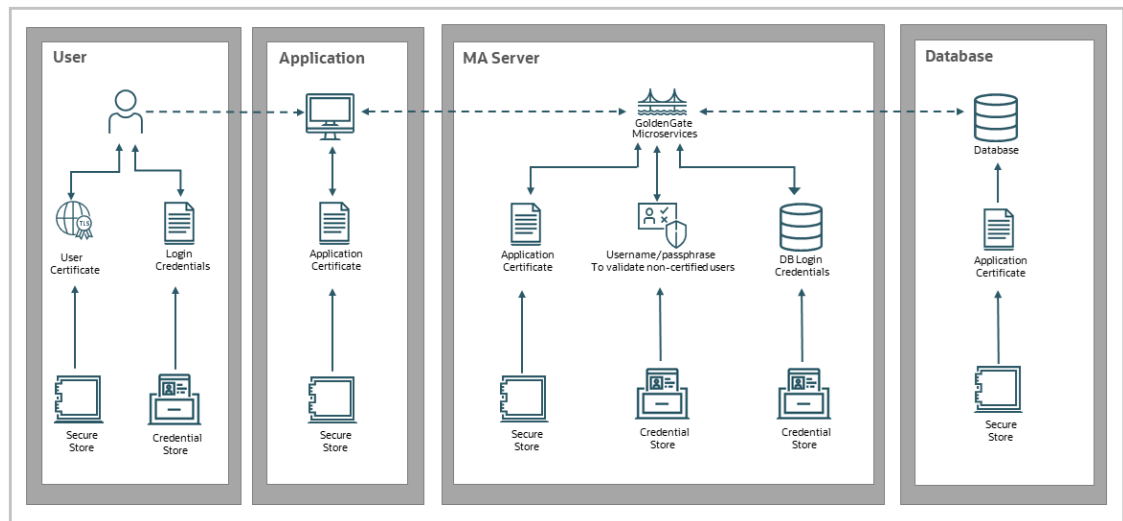
Authorization (AZ) includes Oracle GoldenGate to Oracle GoldenGate communication, and tasks for network and server configuration.

All the security configurations and services are common to MA-based servers. These servers authenticate, authorize, and secure access to command and control, monitoring, data conveyance, and information service interfaces for Oracle GoldenGate.

Oracle GoldenGate Microservices provides an infrastructure for building service-aware applications to operate and integrate into global, cloud-based deployment environments

Oracle GoldenGate Authentication

The goal of an authenticated identity in Oracle GoldenGate is to establish identity authentication between users, applications, and Oracle GoldenGate Microservices deployments. The authentication design provides the option to either validate users and applications with certificates or user credentials stored as credential aliases (username and password pair).



Authentication with User Credentials

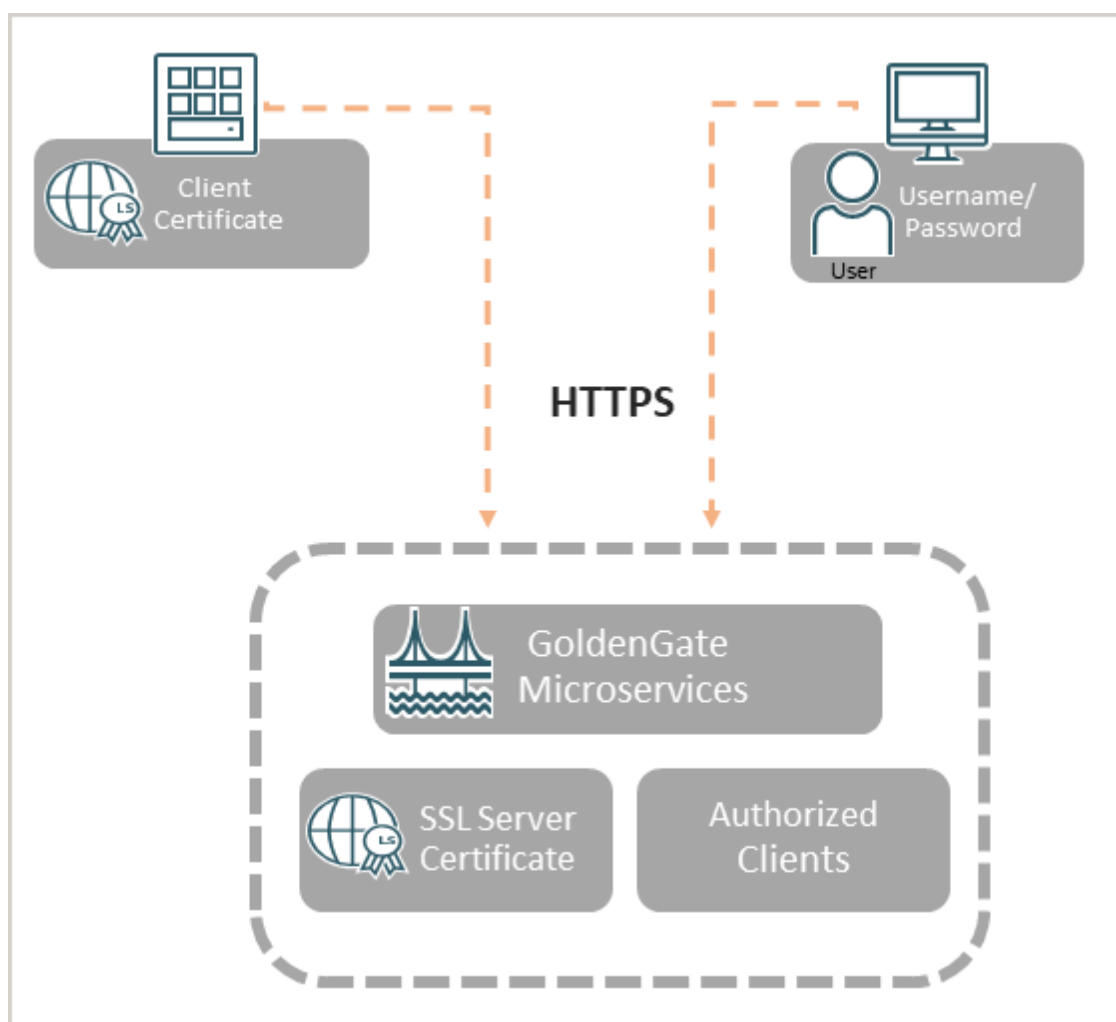
User credentials (username, password) combination is stored in the credential store and is accessible using credential aliases, allowing users to keep the user ID and password ensconced in the credential store and only the alias to connect to an Oracle GoldenGate deployment or another application.

Strong password policy is implemented with guidelines to have 1 uppercase, 1 lowercase, 1 numeric, and 1 special character.

The maximum password length in Oracle GoldenGate 23ai and higher releases is 1024 bytes.

Certificate Management

When a client attempts to connect to a source or target database through a TCPS (Secure TCP) database connection service, Oracle GoldenGate uses TLS certificate-based authentication to verify the connection, as shown in the following diagram.



Notice that secure network communication within Oracle GoldenGate is done with TLS (HTTPS underneath REST-API calls and WSS for the Distribution Path).

Oracle GoldenGate supports mutual TLS (mTLS). See the [Secure Communication Using TLS and mTLS Support](#) to learn about how mTLS can be used as an additional layer of authentication during the client-server handshake.

You can create self-signed client and server certificates or you can install CA-signed server-side certificates for authentication. These certificates can be managed for checking certificate validity, expiry, and usage from Oracle GoldenGate Service Manager.

To learn about managing certificates from the Service Manager web interface, see [Manage Certificates for Deployments](#).

For steps to create certificates, see [Create Certificates for Secure Deployments](#).

To learn about implementing external rootCA certificates for connecting secure deployments host of different servers, see [Connecting Two Deployments Using External RootCA Certificate](#)

Authentication with Certificates

User authentication can be done with certificates when connecting between deployments. This authentication method is an alternative to using credential aliases (user ID, password) for authentication.

You can set up Certificate authentication when configuring a distribution path to connect the source deployment to a target deployment that is on a different network. This scenario is most common different organizations need to communicate over a secure network for data transfer. In this case, a distribution client (distclient) user with Operator role is created on the target deployment and the client certificate is stored on the source deployment. The **distclient** user presents the client certificate when connecting to the source deployment. This certificate is verified by the trusted rootCA certificate available on the target deployment and m-TLS is used by the source to validate the target server certificate. To learn more about setting up certificates for user authentication, see the [Connecting Two Deployments Using External RootCA Certificate](#) and [Connecting Two Deployments Using External RootCA Certificate](#).

Also see the [Add a Distribution Path](#) section for steps used when configuring certificates for two separate deployments.

Authorization in Oracle GoldenGate

Authorization in Oracle GoldenGate relies on user roles. Actions performed in Oracle GoldenGate depend on the user role applied to a user. Use the [REST API Service Endpoints](#) for details on determining the required user role for performing different actions.

You can choose and assign from the following user roles when creating Oracle GoldenGate users.

Table 4-1 Oracle GoldenGate User Roles and Privileges

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 server.
Administrator	Grants full access to the user, including the ability to alter general, non-security related operational parameters and profiles of the server.
Security	Grants administration of security related objects and invoke security related service requests. This role has full privileges.

Oracle GoldenGate User Management

Oracle GoldenGate users that are created at different stages of deployment configuration, have different access functions for Oracle GoldenGate Microservices.

The *first* user for Oracle GoldenGate is set up from the Oracle GoldenGate Configuration (OGGCA) utility. This user can access the Service Manager and all Microservices for all deployments associated with the Service Manager on the host.

After you access the Service Manager using the first user, you can create users from the Administrator page of the Service Manager. Depending on the user role, you can enable users to access Oracle GoldenGate Microservices.

Here are some examples of accessing Oracle GoldenGate Microservices with different user roles:

- Example 1: If you create a user with the Operator role in Administration Service, you cannot access the Service Manager with these credentials.
- Example 2: If you create a user with the Operator role in the Service Manager, you cannot access the Administration Service with these credentials.

Database Authentication and Authorization

Learn about database authentication and authorization configurations required when setting up the database to work with Oracle GoldenGate.

Database Authentication

Database connections are managed with the USERIDALIAS parameter in Oracle GoldenGate. You cannot directly connect to the database from Oracle GoldenGate. The DBLOGIN USERIDALIAS, FETCHUSERIDALIAS, and MINNINGDBUSERIDALIAS contains username and password for any supported database, which is stored in Oracle GoldenGate credentialstore.

Kerberos Authentication

Oracle GoldenGate supports operating system level login for Oracle database. The support of Kerberos authentication is enabled on top of the existing OS level, as an external authentication feature.

For implementation details regarding Kerberos Authentication, see [Configure Kerberos Authentication](#).

Secure Data at Rest

All customer related data such as trail files as well as any dependent/derived data such as Bounded Recovery, spilled/ staged out data on disk is maintained in an encrypted format in the Oracle GoldenGate environment.

Oracle GoldenGate uses different encryption techniques to secure data at rest. Encryption is done using a master key and supports the use of multiple master keys.

Trail File Encryption

Oracle GoldenGate provides AES-based methods for trail file encryption. Features for trail file encryption include:

- ANSI X9.102 is the key wrap algorithm used for encapsulation encryption.
- Data encryption key (DEK) for each trail file (Local key) is included.
- An encrypted version of the local key is included in the trail file header, and a master key is used to encrypt the data encryption key.
- Encryption levels are AES128, AES192, AES 256.

Oracle Key Management Services

Oracle GoldenGate offers the following methods for key management:

- **Local Wallet:** The encryption master key is stored in the local wallet file.
- **Oracle Key Vault (OKV):** The encryption master key is stored in Oracle Key Vault. This Oracle Key Vault service can reside on a different server than Oracle GoldenGate.

The OKV method is highly recommended for on-premise Oracle GoldenGate trail encryption. This method is available with Oracle GoldenGate Microservices Architecture and requires defining an encryption profile in Oracle GoldenGate. See [Using Oracle Key Vault Trail File Encryption in Oracle GoldenGate](#).

- **Oracle Cloud Infrastructure Key Management Service (OCI KMS):** The encryption master key is stored in OCI KMS. Master key never leaves OCI KMS. This method is recommended if your Oracle GoldenGate deployment can access OCI KMS.

This method works with Oracle GoldenGate Microservices Architecture and requires defining an encryption profile in Oracle GoldenGate. See [Configure Oracle GoldenGate Processes to Enable OCI KMS Trail File Encryption](#).

Why Use KMS to Store Oracle GoldenGate Encryption Keys?

Oracle GoldenGate encryption of trail files is enhanced by using OKV or OCI KMS as the Key Management Service (KMS) to store master keys.

Each time Oracle GoldenGate creates a trail file, it generates a new encryption key automatically. This encryption key encrypts the trail contents. The master key encrypts the encryption key. This process of encrypting encryption keys is known as **key wrap** and is described in standard ANSI X9.102 from American Standards Committee.

Key management refers to managing cryptographic keys within an enterprise. It deals with generating, exchanging, storing, using, and replacing keys as required. A KMS also includes key servers, user procedures, and protocols. The security of the enterprise is dependent upon successful key management.

The advantages of using KMS with Oracle GoldenGate are:

- Centralized lifecycle management of master keys. You'll be able to generate and upload master keys to Oracle Key Vault directly using custom attributes and perform lifecycle maintenance tasks within the KMS directly.
- Oracle GoldenGate doesn't need to store the master keys locally and is not involved in the lifecycle management of the master keys.
- Oracle GoldenGate can leverage from the specialized KMS features that provide key management with several layers of security.

Encryption Support for Cached Files

Starting with Oracle GoldenGate 23ai, outsourced user data such as cache files are encrypted with the same Encryption Algorithm used for the trail files.

Encryption Support for Persisted BR Files

Starting with Oracle GoldenGate 23ai, outsourced user data such as Bounded Recovery (BR) persisted data files (PCDFs) are encrypted with the same Encryption Algorithm used for the trail files.

Secure Data in Transit

Oracle GoldenGate protects data in transit or data in motion using the following features:

- Streaming protocols with native HTTP authorization is supported. It includes a header in the initial WebSocket establishment request, which is the secure web socket (WSS) protocol. The Oracle GoldenGate server checks the authorization header to approve or

deny the request based on whether the role associated with the requesting user is equal to or greater than the role assigned for Web Sockets establishment requests. Oracle GoldenGate supports TLS 1.2 and TLS 1.3 authentication. You can choose between different TLS encryption options and apply the relevant cipher suites.

- Target-initiated Paths support data transition when working in Demilitarized Zones (DMZ).

Also see [Distribution Path Streaming Protocols](#).

Secure Communication Using TLS and mTLS Support

The Transport Layer Security (TLS) protocol provides secure communication over a network by ensuring secure data transmission. Data exchange between the server (GoldenGate instance) and the clients including GoldenGate web interface, AdminClient, and other applications, is authenticated and encrypted. It uses cryptographic algorithms to encrypt data, preventing unauthorized parties from accessing or modifying the information being transmitted.

TLS verifies the server's identity to ensure the communication is with the intended party and not a malicious actor. HTTPS and WSS use the TLS protocol. HTTPS requests are made through REST-API calls and WSS is used for the secure routing of trail files by the Distribution Path.

Mutual TLS (mTLS) is an extension of the TLS protocol that adds an extra layer of security by authenticating both the client and the server in a communication session. It is an enhancement to traditional TLS, providing a two-way trust, and is an additional defense against impersonated attacks. During the client-server handshake, the *server* presents its digital certificate to prove its identity to the *client* so that the client verifies the server's identity. The *client* also presents its digital certificate to prove its identity to the *server* so that the server verifies the client's identity.

In a secure deployment, mTLS is used by the Distribution Path process to send trail files using the WSS protocol in either one of the following cases:

- Sending trail data from the Distribution Service to the Receiver Service
- Sending trail data using the Target Initiated Distribution Path, where the Receiver Service requests the Distribution Service to send trail files over the network.

Using the Distribution Client (DistClient) authentication by presenting the DistClient digital certificate, enables secure network communication using mTLS.

A secure deployment is recommended for Oracle GoldenGate. It requires the server's certificate, private key, and the signing Certificate from the Certificate Authority (CA certificate). The DistClient certificate, private key, and the signing certificate from the Certificate Authority (CA certificate) can be applied at any time.

However, NGINX could be used with unsecure deployment to still provide TLS support for distribution path. The TLS will be terminated at NGINX. The NGINX certificate will be verified by the client. However, the client certificate is optional by Oracle GoldenGate default NGINX configuration.

See [Target-initiated Path](#) and [Oracle GoldenGate Reverse Proxy Support](#) for details.

Note

mTLS is not supported when a Reverse Proxy is used.

How Transport Layer Security Works in an Oracle Environment: The TLS Handshake

When a network connection over Transport Layer Security is initiated, the client and server perform a TLS handshake before authentication.

The handshake process is as follows:

1. The client and Oracle GoldenGate instance establish the cipher suites and the encryption algorithms used for data transfers.
2. The Oracle GoldenGate instance sends its certificate to the client, and the client verifies that the Oracle GoldenGate instance's certificate was signed by a trusted CA. This step verifies the identity of the Oracle GoldenGate instance.
3. Similarly, if client authentication is required, then the client sends its certificate to the Oracle GoldenGate instance. The Oracle GoldenGate instance verifies that the client's certificate was signed by a trusted CA.
4. The client and Oracle GoldenGate instance exchange key information using public key cryptography. Based on this information, each generates a session key. A key is shared by at least two parties (usually a client and an Oracle GoldenGate instance) that is used for data encryption for the duration of a single communication session. Session keys are typically used to encrypt network traffic. A client and an Oracle GoldenGate instance can negotiate a session key at the beginning of a session, and that key is used to encrypt all network traffic between the parties for that session. If the client and Oracle GoldenGate instance communicate again in a new session, then they negotiate a new session key. All subsequent communication between the client and the Oracle GoldenGate instance is encrypted and decrypted using this session key and the negotiated cipher suite.

The authentication process is as follows:

1. On a client, the user initiates a network connection to Oracle GoldenGate by using TLS.
2. TLS performs the handshake between the client and Oracle GoldenGate.
3. If the handshake is successful, then the GoldenGate instance verifies that the user has the appropriate authorization to access the Oracle GoldenGate instance.

How Does Oracle GoldenGate Use Transport Layer Security for Authentication

Using Oracle GoldenGate TLS functionality to secure communications between Oracle GoldenGate instances and clients, you can do the following:

- Use TLS to encrypt the connection between Oracle GoldenGate instances and clients
- Authenticate any client or any other Oracle GoldenGate instance, to any other Oracle GoldenGate instance that is configured to communicate over TLS.

After TLS is established, the Oracle GoldenGate instance will need to authenticate the client to make sure it has correct privileges to send trail. The following authentication modes are used:

- The Oracle GoldenGate Instance will authenticate the client using the **user ID** and **password** that has at least the **Operator** role on the target side. The user ID and password is stored as alias in the client side. The alias is used when creating the distribution path.

See [Connecting Two Deployments Using External RootCA Certificate](#) to learn about setting up an Operator role user and then use this user ID, password alias when creating a distribution path connection between remote deployments.

- The Oracle GoldenGate instance will use the **client certificate** to authenticate the client request, assuming that there is a corresponding certificate user created on the target side.

To steps to create certificates for the client and server sides, see [Create Certificates for Secure Deployments](#).

- The Oracle GoldenGate instance will authenticate the client using **OAuth**, when it uses external identification provider (IdP) such as **IAM**. See [Delegate User Management to an External Identity Provider](#).

TLS Cipher Suites

A cipher suite specifies one algorithm for each of the following tasks:

- Key exchange
- Bulk encryption
- Message authentication code (MAC)

The default cipher suites for TLS 1.2 and 1.3 are supported except for the weak ciphers.

Oracle strongly recommends that you use TLS 1.3 to meet your security requirements.

The default cipher suites for TLS 1.2 are supported except for the weak ciphers.

Target-initiated 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 host 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 to pull the requested trail files. The Receiver Service endpoints display that the retrieval of the trail files was initiated by the Receiver Service.

A path (read-only) is created from the Receiver Service of the target deployment and has the property `TARGET_INITIATED`. This path is accessible from the Distribution Service also. The path information is stored on the target system. If the communication is lost, then the Receiver Service on the target host needs the path definition to restart the connection. This information is shared with the Distribution Service when the path is running. The path is **ephemeral** on the source deployment. Ephemeral paths help with consolidation of path configuration and with reinforcement of target-to-source connection initiation.

When the path is stopped or disconnected, the Distribution Service removes all the path information including the path definition. However, the checkpoint file is retained because the checkpoint is used to decide whether old trails can be purged or not. It is recommended that old trails are not purged unless the path is intentionally deleted.

Related Topics:

- [Authentication with Certificates](#)
- [Connecting Two Deployments Using External RootCA Certificate](#)
- [Add a Distribution Path](#)

Oracle GoldenGate Reverse Proxy Support

Reverse proxy enables accessing microservices using one single port (443) in a deployment. This enables encapsulation of the URL for microservices over an unsecure deployment.

Note

Reverse proxy is optional, however, Oracle recommends that you ensure easy access to microservices and provide enhanced security.

You can run microservices in an unsecure deployment on loopback address and front it with an HTTP reverse proxy using the Nginx installation.

Oracle GoldenGate MA includes a script called `ReverseProxySettings` that generates configuration file for only the Nginx reverse proxy server.

For example, the Administration Service is available on `http://goldengate.example.com:9001` and the Distribution Service is on `http://goldengate.example.com:9002`. With reverse proxy, each of the microservices can simply be accessed from the single address. For example, `http://goldengate.example.com/distsrvr` for the Distribution Service. The URL is different for each service and is by name instead of by port.

These values are used when connecting to the Service Manager and are required when authentication is enabled.

See [Configure Oracle GoldenGate Reverse Proxy with NGINX](#) for implementing NGINX reverse proxy with Oracle GoldenGate.

Accountability

Oracle GoldenGate includes functional security capabilities such as REST API call auditing and system event logging.

Auditing and Logging

In Oracle GoldenGate, auditing is enabled by default. The following auditing features are available with Oracle GoldenGate:

- You have the `$OGG_HOME/lib/utl/logging/ogg-audit.xml` file that defines the audit conditions.
- By default, the `syslog` appender is used, which writes the audit into system security log. The system log messages are stored in the `/var/log/secure` directory.

Note

System-level events are logged to `/var/log/secure`, while Oracle GoldenGate-specific logs are written to `$OGG_VAR_HOME/log`.

- As the `ogg-audit.xml` file is in the software directory, it manages all deployments for this software.

You can copy this file into \$OGG_ETC_HOME/config/logging and modify the file. It is only specific to the Oracle GoldenGate deployment.

- You can configure logging for all deployments in the installation directory \$OGG_HOME/lib/utl/logging.
- You can configure logging for a specific deployment in the installation directory \$OGG_ETC_HOME/etc/conf/logging.

Structure of the Log Files

The structure of log files includes the following information:

- Date
- Info | Warning | Error
- Message

Service Manager Deployment Logs

The Service Manager deployment logs include the following log files:

- ServiceManager.log
- AdminClient.log
- RESTAPI.log
- Pluggingssvr.log (only if the Plugging Service is installed).

User Deployment Logs

The user deployment logs include the following files:

- adminssvr.log
- distssvr.log
- recvsrvr.log
- pmsrvr.log
- extract.log
- replicat.log
- ER-events.log
- ggserr.log
- adminclient.log
- restapi.log

Miscellaneous

Some additional security features available with Oracle GoldenGate are mentioned in this section.

FIPS 140-2

Federal Information Processing Standards (FIPS) are standards and guidelines for federal computer systems that are developed by the U.S. National Institute of Standards and Technology (NIST).

Oracle GoldenGate is FIPS-140-2 Level 1 compliant. When FIPS 140-2 settings are configured for the Oracle GoldenGate, the Oracle GoldenGate instance uses FIPS 140-2 Level 1 validated cryptographic libraries to protect data at rest and in transit over the network. Oracle GoldenGate currently uses the OpenSSL FIPS Provider as the FIPS 140-2 level 1 validated cryptography library.

Note

Note that Oracle GoldenGate FIPS settings enforce the use of FIPS-approved algorithms for the Oracle GoldenGate only. Third-party vendor software used with Oracle GoldenGate running in FIPS mode must use only these FIPS-approved algorithms, or else the vendor software will encounter failures.

FIPS was developed in accordance with the Federal Information Security Management Act (FISMA). Although FIPS was developed for use by the federal government, many private sector entities voluntarily use these standards.

FIPS 140-2 specifies the security requirements that will be satisfied by a cryptographic module, providing four increasing, qualitative levels intended to cover a range of potential applications and environments. Security Level 1 conforms to the FIPS 140-2 algorithms, key sizes, integrity checks, and other requirements that are imposed by the regulations. FIPS 140-2 Security Level 1 requires no physical security mechanisms in the module beyond the requirement for production-grade equipment. As a result, this level allows software cryptographic functions to be performed in a general-purpose computer running on a specified operating environment.

Oracle GoldenGate Security Feature: Implementation

This section is relevant for DBAs and developers engaged in security configurations. It describes the implementation steps of security features in Oracle GoldenGate Microservices Architecture deployment and Service Manager.

The following steps describe the Oracle GoldenGate security features that you can implement when building a secure Oracle GoldenGate environment.

1. Use the **latest software version**.

Download latest Oracle GoldenGate software. Make sure you're using the most recent Release Update (RU). Installing the most recent software can often improve security as newer versions frequently include fixes and updates that address vulnerabilities discovered in previous versions.

2. If you use OGGCA to deploy Oracle GoldenGate, **ensure that security is enabled**. A secure deployment consists of the Server certificate, its Private Key, and the CA-signed (root) certificate. This enables secure communication using TLS for REST-API.

If you are unable to implement Oracle GoldenGate with the Server certificate, you may consider using the **Reverse Proxy for secure network communication**. In this scenario, make sure the server is locked down.

3. Use passwords that adhere to the **Strong Password Policy**. You might utilize a local Oracle GoldenGate Administration User for the Service Manager and an external identity Provider (IDP) for external authentication using OAuth2/OIDC. You can use Oracle Cloud-based OCI-IAM, IDCS, or On-Premise OAM as IDP providers. IDP allows you to take advantage of SSO, MFA, or Token-based Authentication.
4. Oracle GoldenGate supports role-based authentication control (RBAC) for user authorization. Use the least privilege best practices for Oracle GoldenGate users.
5. Verify that **encryption is applied to the trail files**. The trail file encryption master key is managed locally. Key management systems like OCI-KMS or OKV, which are external, offer an even higher degree of protection.
6. Oracle GoldenGate utilizes a **PKCS#12** wallet to hold Oracle GoldenGate and database credentials. Only a USERIDALIAS can be used to establish database connections. You might want to update the Oracle GoldenGate Administrator user's credentials.
7. For secure network communication to the database, use the provided database secured network client. The Oracle Database Server provides TCPS and native network Encryption (NNE).
8. For the Distribution path protocol, use the Secure WebSocket Protocol (WSS). To protect Oracle GoldenGate DistPath, you can utilize Credentials, Client certificates (Mutual TLS), or OAuth/OIDC.
9. In firewall-secured DMZ environments, use Target Initiated DistPath. This makes it possible for the target Receiver Service to start the DistPath.
10. Federal Information Processing Standards (FIPS) compliance is another feature of Oracle GoldenGate. During deployment, the additional security standard (FIPS) can be enabled.

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.

Create Certificates for Secure Deployments

Learn about creating different types of certificates for a deployment in a hub or external certificate when transporting trail data across two different host deployments.

Each system (deployment) has its own set of root, server, and client certificates.

The most common use case for generating certificates within the same organization is to create a single trusted root certificate (rootCA), which is used at different locations but within the same organization.

The other use case is where a secure deployment must be provided between fully independent organizations in which even the trusted root certificates (rootCA01, rootCA02) are different. Such a case is more complex and is described in the [Connecting Two Deployments Using External RootCA Certificate](#).

In this section, you will learn to create trusted certificates (rootCA), server certificates, client certificates, and distribution client certificate for secure Oracle GoldenGate Microservices Architecture deployments, when setting up a secure deployment or authenticate connections between two separate deployments.

Note

The provided OpenSSL commands are using a self-signed certificate. This case is only used to demonstrate how to set up a secure environment with non-commercial certificates. For secure environments, it is recommended to use certificates provided by commercial providers.

To know more about managing certificates from the Oracle GoldenGate web interface, see [Certificate Management](#).

Create RootCA and Server Certificates

Various client and server certificates may be required for a deployment. To create a trusted root CA and Server certificate for a host deployment, use the commands described in the following sections.

The commands used to generate these certificates are OpenSSL commands.

In the following example, the deployment is done on the host *west01.dc1.example.com* within the fully qualified domain name *dc1.example.com*. If you create multiple Oracle GoldenGate instances on different servers, you might replace the *server01* with your hostname and replace the qualified domain name accordingly.

Create Trusted RootCA Certificates

Generate a Trusted RootCA Certificate using the following commands:

```
openssl req -subj "/CN=RootCA" \
            -newkey rsa:2048 \
            -nodes \
            -keyout rootCA_key.pem \
            -new -x509 -days 365 \
            -out rootCA_cert.pem
```

This command creates two files with the root certificate *rootCA_cert.pem* and the private key *rootCA_key.pem*. Both files are stored in the Privacy Exchanged Mail (PEM) format. The private key is created in a Public-Key Cryptography Standards (PKCS) #8 format. The root certificate *rootCA_cert.pem* is used within a secure Oracle GoldenGate deployment for the server certificate. You can also add the distribution client (distclient) certificate within the deployment. Here, the root certificate is used again.

Create Server Certificates

With the rootCA certificate and private key, you can create the server certificate. Use a configuration file similar to the following for `west01_cert.cnf`:

```
extendedKeyUsage = serverAuth
subjectAltName =
DNS:west01,DNS:west01.dc1.example.com,DNS:localhost,IP:127.0.0.1
```

After creating the configuration file for the Server certificate, the following commands are used to create the server certificate and the private key files:

```
subject="/C=US/O=OGG example/CN=west01"

openssl req -subj "${subject}" \
    -newkey rsa:2048 -nodes \
    -keyout west01_key.pem \
    -new \
    -out west01.csr

openssl x509 -CAcreateserial \
    -CA rootCA_cert.pem \
    -CAkey rootCA_key.pem \
    -req \
    -in west01.csr \
    -extfile west01_cert.cnf \
    -days 365 \
    -out west01_cert.pem
```

Both files are stored in the Privacy Enhanced Mail (PEM) and the private key is created in a Public-Key Cryptography Standards (PKCS) #12 format. The server certificate and server private key are used within the Oracle GoldenGate deployment. The Common Name (CN) within the subject is using the hostname `west01` to uniquely identify the server.

Create External Trusted RootCA and Distribution Client Certificates

If you are using Certificates as the target authentication method in scenarios where two deployments need to connect the distribution and receiver paths to transfer trail data, then you will need to use an **external RootCA** certificate to validate a distribution client user using a distribution client (**distclient**) certificate. The distribution client certificate is stored on the target deployment and a distribution client user (operator role) is created on the source deployment. The validation type of this user is set to Certificate.

Note

Distribution Path created using Certificates require target deployment to have an x-Cert user configured that has username same as Subject DN of Dist Path Certificate. See [About Distribution Service Path](#).

Create a RootCA External Certificate in the Target Deployment

Use the following steps to create and manage the root CA external certificate (rootCA_ext) for a target deployment that is different from the source deployment.

Here is a sample rootCA_ext.cfg configuration file:

```
[ req ]
default_bits = 4096
default_md = sha512
prompt = no
encrypt_key = no
distinguished_name = req_distinguished_name
req_extensions = v3_req
x509_extensions = v3_ca
x509_extensions = usr_cert
[ req_distinguished_name ]
#countryName = "US"
#stateOrProvinceName = "CA"
#localityName = "Redwood City"
#streetAddress = "400 Oracle Pkwy"
#organizationName = "Oracle USA Inc"
#organizationalUnitName = "Security"
commonName = "rootCA_ext"
#emailAddress = "rootsecurity@oracle.com"
[ v3_req ]
basicConstraints=CA:TRUE
[ v3_ca ]
basicConstraints=CA:TRUE
[ usr_cert ]
basicConstraints=CA:TRUE
[ my_extensions ]
```

The command to generate the rootCA external certificate is:

```
openssl req -subj "/CN=RootCA_ext" \
            -newkey rsa:2048 -nodes \
            -keyout rootCA_ext_key.pem \
            -new -x509 \
            -days 365 \
            -out rootCA_ext_cert.pem
```

Create a Distribution Client Certificate

The command to generate a client certificate is similar to the following:

```
extendedKeyUsage = clientAuth
openssl req -subj "/CN=distclient" \
            -newkey rsa:2048 -nodes \
            -keyout distclient_key.pem \
            -new \
            -out distclient.csr
```

This **distclient** certificate is verified by the rootCA_ext certificate, using the following command:

```
openssl x509 -CAcreateserial \
        -CA rootCA_cert.pem \
        -CAkey rootCA_key.pem \
        -req \
        -in distclient.csr \
        -extfile distclient_cert.cnf \
        -days 365 \
        -out distclient_cert.pem
```

The **distclient** certificate and the private key are generated. Both files are stored in the Privacy Exchanged Mail (PEM) and the private key is created in a Public-Key Cryptography Standards (PKCS) #12 format.

Configure Oracle GoldenGate Reverse Proxy with NGINX

Learn how to configure reverse proxy service using NGINX for accessing Oracle GoldenGate Microservices without using port numbers.

Prerequisites for Using ReverseProxySettings

You can use any reverse proxy service with MA. The following example provides a process that you can follow to configure other reverse proxy services in conjunction with the documentation for your proxy server.

The following prerequisites provide details on the minimum requirements to configure an NGINX Reverse Proxy. Similar requirements may be required for your environment and reverse proxy, if you are using a different utility for proxy configuration.

① Note

When installing Oracle GoldenGate 26ai on Oracle Linux 8 or RHEL 8, ensure that the NGINX version is 1.24 or higher by enabling the appropriate NGINX module stream.

1. Install NGINX, see [Install the NGINX Web Server and Proxy on Oracle Linux](#). For Oracle Linux, the command to install NGINX is:

```
yum -y install nginx
```
2. Check the JRE version to be JRE 8 or higher.
3. Install Oracle GoldenGate MA.
4. Create one or more active MA deployments.
5. Ensure that the Oracle user has sudo permissions.
6. Configure the PATH environment variable to include the NGINX installation directory path.

Run the ReverseProxySettings Utility to Configure NGINX

An Oracle GoldenGate Microservices Architecture installation includes the ReverseProxySettings utility. The ReverseProxySettings utility is located in the `$OGG_HOME/lib/utl/reverseproxy` directory.

To identify additional commands that can be used with the ReverseProxySettings utility, run the utility with the `--help` option:

```
$OGG_HOME/lib/utl/reverseproxy/ReverseProxySettings --help
```

Options available with the ReverseProxySettings utility are:

-o or --output

The output file name. The default file name is `ogg.conf`.

-P or --password

A password for a Service Manager account.

-l or --log

Log file name and initiates logging. The default is no logging.

--trailOnly

Configure only for inbound trail data.

-t or --type

The proxy server type. The default is Nginx.

-s or --no-ssl

Configure without SSL.

-h or --host

The virtual host name for reverse proxy.

-p or --port

The reverse proxy port number. The defaults are 80 or 443.

-? or --help

Display usage information.

-u or --user

Name of the Service Manager account to use.

-v or --version

Displays the version.

Run the ReverseProxySettings Utility

To use the ReverseProxySettings utility:

1. To generate a configuration file for NGINX reverse proxy, navigate to the location of the ReverseProxySettings utility:

```
cd $OGG_HOME/lib/utl/reverseproxy
```

2. Run the ReverseProxySetting utility:

```
ReverseProxySettings -u adminuser -P adminpwd -o ogg.conf http://localhost:9100
```

In this code snippet, *adminuser* is the deployment user name and *adminpwd* is the deployment user password used to login to the deployment.

3. Copy the ogg.conf file generated in step 2 to /etc/nginx/conf.d directory.

```
sudo cp ogg.conf /etc/nginx/conf.d/.
```

4. Update the ogg.conf file for SSL/TLS settings using the following steps:

- a. Find ssl_ciphers line, replace it with the following:

```
ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-CHACHA20-POLY1305;
```

- b. Add another line below this line:

```
ssl_conf_command Ciphersuites
TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256;
```

- c. If you are using a self-signed certificate, make sure that you have the private key and the certificate pem file copied to /etc/nginx directory, as shown in the following sample:

```
sudo cp ogg.key /etc/nginx/.
sudo cp ogg.pem /etc/nginx/.
```

Change the following two lines to point to the correct file location:

```
ssl_certificate      /etc/nginx/ogg.pem;
ssl_certificate_key  /etc/nginx/ogg.key;
```

- d. If you are using a root CA signed certificate, make sure that you have all the intermediate CA certificates added to your server certificate:

```
cat server_cert.pem intermediate_ca_cert1.pem intermediate_ca_cert2.pem
> ogg_chain.pem
```

Copy the private key and the certificate chain to /etc/nginx:

```
sudo cp ogg_chain.pem /etc/nginx/.
sudo cp ogg_key /etc/nginx/.
```

Change the following two lines to point to the correct file location:

```
ssl_certificate      /etc/nginx/ogg_chain.pem;
ssl_certificate_key  /etc/nginx/ogg.key;
```

5. Validate the NGINX configuration:

```
sudo nginx -t
```

The output would show the following, if the command is successful:

```
NGINX: the configuration file /etc/NGINX/NGINX.conf syntax is ok
NGINX: configuration file /etc/NGINX/NGINX.conf test is successful
```

6. Restart NGINX service:

```
sudo systemctl restart nginx
```

If the changes for the configuration file are not loaded, stop and restart the proxy.

7. To test if you can access the microservices after NGINX is set up successfully, open the web browser.
8. Enter the proxy URL for the Service Manager using port number 443, similar to the following:

`http://dc.example.com:443`

This would open the Service Manager login page, from where you can access the other microservices also. If you want to directly access a microservice, you can enter the proxy URL for that microservice, as given in the `ogg.conf` file, generated previously.

Also see this [video](#) on configuring the NGINX reverse proxy.

SSL Termination

When there is an unsecure connection between the reverse proxy, which uses a TLS-based connection, and the origin server, it is referred to as reverse proxy SSL-termination.

Note

In SSL-Termination the connections between the reverse proxy and the origin servers are unsecure.

However, SSL-bridging is also supported where the connections between the client and reverse proxy is secured and the connection between the reverse proxy and the origin server is also secured.

Note

mTLS is not supported when a Reverse Proxy is used.

Encrypting Trail Files

Learn about using different Oracle key management systems available with Oracle GoldenGate.

Using the Plugin Service for Trail File Encryption

Learn about the prerequisites, requirements, and steps to configure a third-party plugin service for encryption and create an encryption profile in Oracle GoldenGate to allow trail file encryption using the configured plugin.

Create and Apply Encryption Profile in a Deployment

In Oracle GoldenGate, the encryption profile is used to define, which trail encryption method to use.

An encryption profile is the configuration information that is used to retrieve a master key from a **local wallet** or a **Key Management Service (KMS)** such as OKV or OCI KMS. Encryption profile configuration is only available with Microservices Architecture.

Following methods are available for managing encryption of master keys:

- Local Wallets
- Key Management Systems:
 - Oracle Key Vault
 - Oracle Cloud Infrastructure

Each Extract and Replicat process is associated with an encryption profile. The default encryption profile is stored in the **local wallet**, if you haven't specified any other encryption profile.

If you use a different encryption profile, which uses a KMS, then it includes all the information necessary to connect and authenticate to the KMS server. It also contains the details necessary to retrieve a particular master key that will be used for encryption and decryption. Any KMS uses an authentication token to access their APIs. Oracle GoldenGate Microservices Architecture stores this access token as a credential. This credential is created using the encryption profile in Microservices Architecture.

Oracle Golden Gate processes need to make a request to the Key Management Service (KMS) each time a trail file is opened.

- For Oracle Key Vault (OKV), the encryption profile parameter time to live (TTL) is used to keep the master key on memory until TTL has been reached.
- In OCI KMS, the actual master key is never returned and instead the client sends the data to encrypt or decrypt. Thereafter, the server returns the result to the client.

An encryption profile is used by the Oracle GoldenGate processes to encrypt or decrypt depending on whether the processes are writing or reading trail files.

- Extract: Encrypt (writer)
- Replicat: Decrypt (Reader)
- Distribution Service Path (DISTPATH): Encrypt/Decrypt (Writer/Reader).
- LogDump: Decrypt (Reader)

Requirements for Setting up an Encryption Profile

This topic describes the requirements when configuring an encryption profile in Oracle GoldenGate.

You can create multiple encryption profiles within a deployment, but an Oracle GoldenGate process (Extract, Replicat, distribution path) can only use one encryption profile at a time. For distribution paths using filtering, decryption is done to apply the filters but the output trail file remains encrypted. In PASSTHRU, a distribution path will not attempt to use the encryption profile or decrypt the trail file unless explicitly specified.

Any of the existing encryption profiles within a deployment can be set as the default profile. This default profile is only relevant during the creation of an Extract, Replicat or Distribution Path processes. If an encryption profile is not explicitly specified during the creation of a process, the current default profile is assigned to the new process. Changing the default profile does not update the encryption profile assigned to any existing Oracle GoldenGate processes.

Note

It is advised not to change the encryption profile or master key of a process that has already processed trail files.

The Administration Service web interface allows you to manage your encryption profiles. You cannot modify an encryption profile. If you need to change it, you must delete and add a new profile using the Administration Service.

You can configure encryption profiles from the Administration Service or the Admin Client.

Tool to Set up Encryption Profile	Description
Administration Service	To configure the encryption profile using the Administration Server, see Configure an Encryption Profile .
Admin Client	The Admin Client commands used to set up the encryption profile for Extract, Replicat, and Distribution Path, include: <pre>ADD ENCRYPTIONPROFILE, ALTER ENCRYPTIONPROFILE, DELETE ENCRYPTIONPROFILE, INFO ENCRYPTIONPROFILE.</pre> In addition, the ADD or ALTER the Extract, DISTPATH, or Replicat commands have been modified to include the parameter <code>ENCRYPTIONPROFILE encryption-profile-name</code>. To know more, see Admin Client Command Line Interface Commands in <i>Command Line Interface Reference for Oracle GoldenGate</i>.

How to Edit the Local Wallet Encryption Profile

The Local Wallet encryption profile is the default encryption profile for a deployment. You can add a new master key to replace the existing one. Here are the steps to configure the local wallet options:

1. Click Encryption from the left navigation pane of the Administration Service.
2. Select Local Wallet. The Local Wallet Profile page is displayed.
3. To replace the existing master key, click the plus sign next to the Master Keys section. A confirmation box is displayed to replace the current master key.

- Click OK to change the master key.

Configure an Encryption Profile

Oracle GoldenGate Administration Service provides options to set up encryption profiles for Extract and Replicat processes.

Use the following steps to create an encryption profile using Oracle Key Vault (OKV) or OCI Key Management System (OCI KMS) options:

Note

If you are using OCI GoldenGate, then there is another option for creating encryption profiles with and without resource principals. To know more about this option, see Encrypt Trail files in the OCI GoldenGate documentation.

- Click Encryption from the left navigation pane of the Administration Service.
- Click the plus sign + next to OKV or OCI KMS to create an encryption profile for any of these methods. Specify the following details for the OKV configuration:

Option	Description
Profile Name	Name of the encryption profile
Description	Describe the encryption profile.
Default Profile	If you want to make this profile the default, then enable this option.
Encryption Profile Type	Available options are Oracle Key Vault (OKV) and Oracle Cloud Infrastructure (OCI).

- Before you set up OKV, you need to perform a client installation. See Step 1: Configure the Oracle Key Vault Server Environment in the *Oracle Key Vault Administrator's Guide*.

OKV Configuration Options	Options to set up Oracle Key Vault (OKV)
KMS Library Path	Specify the directory location where Oracle Key Vault is installed.
Oracle Key Vault Version	Specify the supported Oracle Key Vault version.
Masterkey Name	Specify the name of the master key
Time to Live	Time to live (TTL) for the key retrieved by Extract from KMS. When encrypting the next trail, Extract checks if TTL has expired. If so, it retrieves the latest version of the master key. The default is 24 hours.

- OCI KMS requires registering of the private/public key (API Signing Key) for accessing the REST API on the Server on which Oracle GoldenGate is deployed. Here are the options to set up the OCI KMS encryption profile.

See [Registering and Managing Keys for OCI KMS](#).

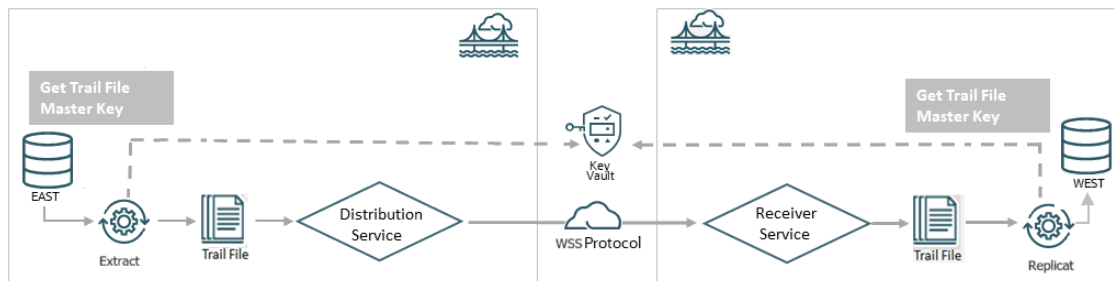
OCI KMS Configuration Options	Options to set up an OCI KMS.

Crypto Endpoint URL	You can access this from the OCI KMS Vault wizard. See OCI Command Line Reference and Managing Keys in <i>OCI Documentation</i> to know more.
Tenancy OCID	When you sign up for Oracle Cloud Infrastructure, Oracle creates a <i>tenancy</i> for your company, which is a secure and isolated partition within Oracle Cloud Infrastructure where you can create, organize, and administer your cloud resources. See Key Concepts in <i>OCI Documentation</i> to learn more.
Key OCID	See the OCI Documentation for details.
User OCID	See the OCI Documentation for details.
API Key	A credential for securing requests to the Oracle Cloud Infrastructure REST API.
API Key Fingerprint	See Required Keys and OCIDs in the OCI documentation for details.

Using Oracle Key Vault Trail File Encryption in Oracle GoldenGate

Learn about the benefits of using Oracle Key Vault (OKV) with Oracle GoldenGate Microservices Architecture. Determine the system requirements, processes and parameters available with Oracle GoldenGate for configuring OKV with Oracle GoldenGate.

The following diagram explains the Oracle Key Vault set up and workflow in the Oracle GoldenGate environment.



As shown in the diagram, you can select the Key Vault (OKV) as the Key Management System while configuring the encryption profile for the deployment in Oracle GoldenGate. This encryption profile is used by the Extract, Replicat processes to store and apply the master keys before transferring trail files using the distribution path and receiving the files on the other end with the RECVPATH process. The Web Secure Socket (wss) protocol is used to validate the connection between the DISTPATH and RECVPATH to establish a secure communication channel.

Oracle Key Vault Capabilities

Oracle GoldenGate 23ai and higher releases support Oracle Key Vault 21.8 for trail file encryption. The following table provides the behavior and capabilities of Oracle Key Vault (OKV).

For more information about configuring OKV, see [Installing and Configuring Oracle Key Vault](#).

KMS Name	KMS Type	Support Tags	Support Importing of Keys
Oracle Key Vault	Keyname and custom attributes for versioning	Yes	Yes

Prerequisites for Configuring GoldenGate Encryption Profile Using Oracle Key Vault

Learn the prerequisites for setting up OKV with Oracle GoldenGate.

In this section, the steps to generate the master key have been performed using the command line. Alternatively, you can use the OKV web interface to create the master key for Oracle GoldenGate. If you want to generate the master key from the OKV web interface, follow the steps provided in Create GoldenGate Master Key in the *Oracle Key Vault Administrator's Guide*.

The following steps are performed on the OKV configuration on the machine where the Oracle GoldenGate instance is running:

1. Download the `okvrestclipackage.zip` from the OKV server, where Oracle GoldenGate is deployed as the same system user as the deployment.
2. Enable the RESTFUL Services using the steps provided in Enabling RESTful Services in *Oracle Key Vault Administrator's Guide*.
3. Set the `OKV_HOME` environment variable.

```
OS> setenv OKV_HOME /u01/app/oracle/OKV
```

The sub-directory structure contains the necessary libraries, binaries, and configuration files for the OKV environment. See [Oracle Key Vault Installation and Configuration](#) in the *Oracle Key Vault Installation and Upgrade Guide* for details about the configuration within the OKV server.

4. Download and install the endpoint file, `okvclient.jar` from the OKV server, where Oracle GoldenGate is deployed as the same system user as the deployment. For example,

```
OS> java -jar okvclient.jar -d /u01/app/oracle/OKV
```

5. Create the key. The name of the wallet is provided by the OKV administrator. The following example show how the key is created:

```
OS> okv managed-object key create
[--activation-date activation date]
[--algorithm algorithm]
[--custom-attribute custom attribute]
[--deactivation-date deactivation date]
[-- extractable extractable]
[--length length]
[--mask mask]
[--name name]
[--wallet wallet]
```

```
ex - okv managed-object key create --algorithm AES --length 256 --mask
```

```
"ENCRYPT,DECRYPT,TRANSLATE_ENCRYPT,TRANSLATE_DECRYPT,TRANSLATE_WRAP,TRANSLATE_UNWRAP" --wallet wallet1
```

Alternatively, you can register your own key, as shown in the following example:

```
OKV> okv managed-object key register
[--activation-date activation date]
[--algorithm algorithm]
[--custom-attribute custom attribute]
[--deactivation-date deactivation date]
[--extractable extractable]
[--length length]
[--mask mask]
[--name name]
--object object
[--wallet wallet]
```

```
ex - bin/okv managed-object key register --length 256 --activation-date
"NOW" --object symkey.txt --algorithm AES --mask
"ENCRYPT,DECRYPT,TRANSLATE_ENCRYPT,TRANSLATE_DECRYPT,TRANSLATE_WRAP,TRANSLATE_UNWRAP"
--wallet wallet1
```

6. Activate the key as shown in the following example:

```
OS> okv managed-object object activate --uuid UUID
```

7. Add the Oracle GoldenGate related key attributes (KeyName, KeyVersion) to the configuration. The key name must match the master keyname in the KMS encryption profile created within Oracle GoldenGate. The key value must match the version number of the masterkey.

```
OS> okv managed-object custom-attribute add --custom-attribute custom
attribute --uuid uuid
```

```
ex - okv managed-object custom-attribute add --custom-attribute
'[ { "name": "x-OGG-KeyName", "value" : "OGG_Masterkey", "type" : "text" },
{ "name": "x-OGG-KeyVersion", "value" : "1", "type" : "text"} ] '
```

8. Use okvutil to list the configuration setting and check the endpoint status. As shown in the following example:

```
OS>okvutil list -v 4
```

The output of this command shows the configuration settings and endpoint status:

```
okvutil version 21.12.0.0.0
Endpoint type: Oracle Database
Configuration file: u01/app/oracle/OKV/conf/okvclient.ora
Server: 100.70.1.1:5696
Standby Servers:
Auto-login wallet found, no password needed
Trying to connect to 100.70.1.1:5696 ...
Connected to 100.70.1.1:5696.
```

Unique ID	Type	Identifier
8BC11E57-5269-403D-9B69-A66641ADAF7	Symmetric	Key -

The next steps are managed within Oracle GoldenGate and are shown as an implementation from the Admin Client.

Client Behavior Against Different Key States for Oracle Key Vault

Following table describes the relative behavior of the of the writer (Extract) or reader (Replicat) client processes depending on the different trail encryption key states.

If the master key is non-extractable, then it implies that OKV cryptographic operations are using remote encryption. This means that the master key cannot leave or be retrieved from OKV. Commands to encrypt and decrypt in Oracle GoldenGate on the basis of once per trail file, are performed inside OKV.

A key can be in the following states:

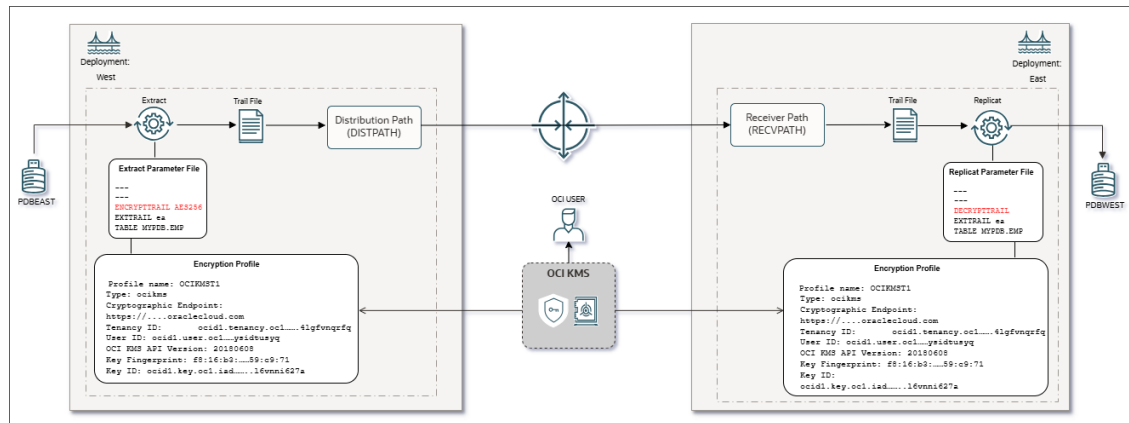
Key State	Trail Writer (encryption)	Trail Reader (decryption)
Active	Trail writer chooses the highest version number with Active state for encryption.	Trail reader can use this key and version number to decrypt the trail.
Preactive	Trail writer ignores and does not consider the key version number with these states.	Not Applicable
Deactivated	None	Trail file reader retrieves and decrypts the trail if the key and version number is deactivated or compromised.
Compromised	None	Trail file reader retrieves and decrypts the trail if the key and version number is deactivated or compromised.
Destroyed	Non	Trail file reader generates an error and abends if the key and version number required to decrypt is in the destroyed or destroyed-compromised state.
Destroyed-Compromised	None	Trail file reader raises an error and abends if the key and version number required to decrypt is in the destroyed or destroyed-compromised state.

Using OCI KMS Trail File Encryption in Oracle GoldenGate

Learn about the prerequisites, requirements, and steps to configure an OCI KMS encryption profile in Oracle GoldenGate to allow trail file encryption using OCI KMS with Extract, Replicat, or Distribution Path processes.

Oracle GoldenGate with OCI KMS Workflow

The following diagrams explains how Oracle GoldenGate works with OCI KMS for trail file encryption.



Setting up OCI KMS encryption profile in Oracle GoldenGate and attaching the encryption profile to Extract. Including the ENCRYPTTRAIL parameter to specify the encryption algorithm in the Extract parameter file.

This diagram shows the source deployment **EAST** containing an Extract associated with the OCI KMS encryption profile. To create the encryption profile in Oracle GoldenGate, the OCI user needs to access the OCI tenancy and get some values from the OCI vault and generate the master key and the API key for the OCI user.

The **OCI vault** contains information about the **tenancy OCID**, **user OCID**, and **cryptographic endpoint URL** used for downloading the digital CA certificate. The **master key** associated with the vault is also generated from the OCI vault. The **API key pair** is also required, which you can generate from the OCI tenancy or upload an existing key pair. See [Configure OCI KMS to Connect with Oracle GoldenGate](#) for details.

After you have saved all the values from the OCI tenancy, you can create an encryption profile in Oracle GoldenGate Administration Service. This encryption profile is then associated with the Extract and the Extract parameter file applies the encryption algorithm (AES 128, AES 192, AES 256) that you have decided to apply, using the ENCRYPTTRAIL parameter. The encrypted trail file is transported by the DISTPATH to the target deployment (WEST). The Replicat parameter file on the target deployment (WEST) includes the DECRYPTTRAIL parameter, which allows decrypting the trail file. See [Configure Oracle GoldenGate Processes to Enable OCI KMS Trail File Encryption](#).

Prerequisites for Connecting Oracle GoldenGate with OCI KMS

Perform the tasks in this section before you begin configuring an OCI KMS encryption profile in Oracle GoldenGate.

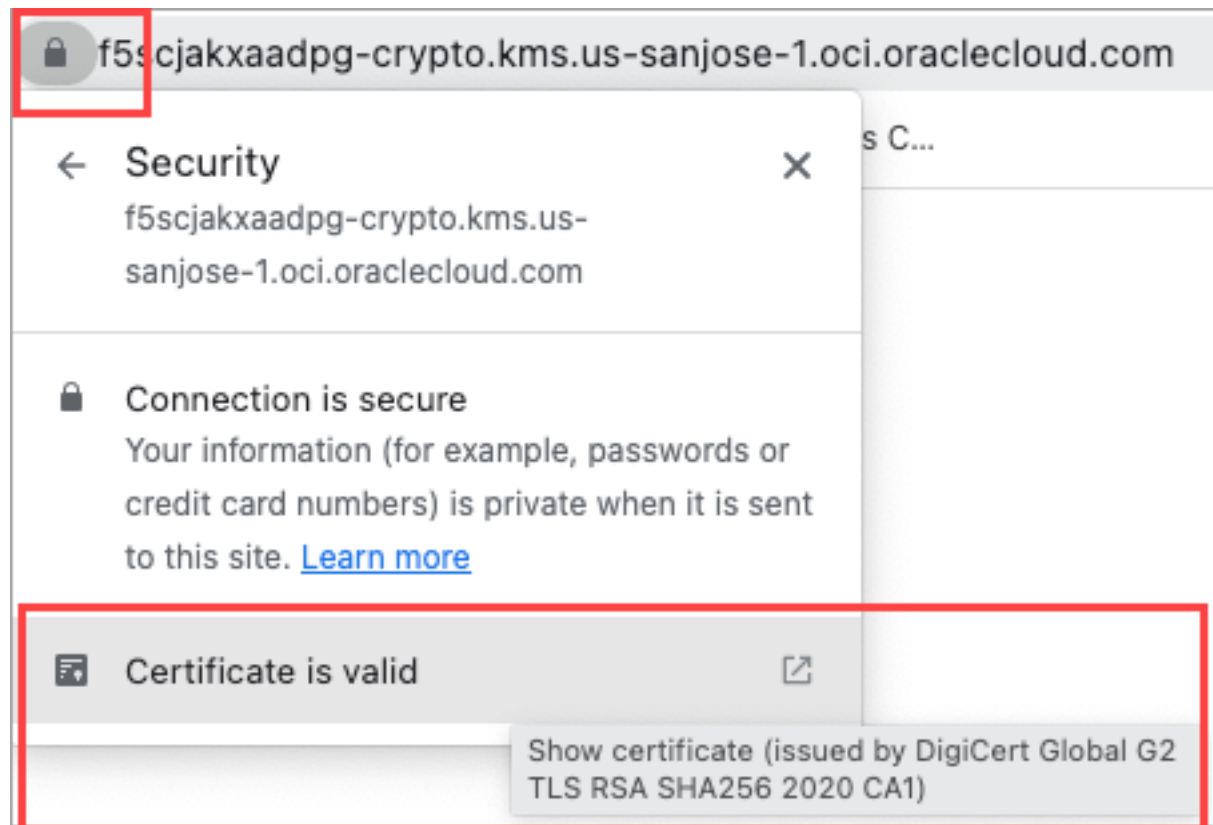
Download the CA Certificate using the Cryptographic Endpoint

To perform the steps in this topic, you need to have a Vault in your OCI tenancy where the cryptographic endpoint URL is mentioned. This URL is required to download the digital CA certificate, for establishing a trusted connection from Oracle GoldenGate to the OCI tenancy. If you don't have an existing vault, then see [Create or Access the OCI Vault](#), and return to this topic for steps to download the CA certificate using the cryptographic endpoint.

If you have an existing Vault in your OCI tenancy, then follow the steps provided in this topic, to download the CA certificate using the cryptographic endpoint.

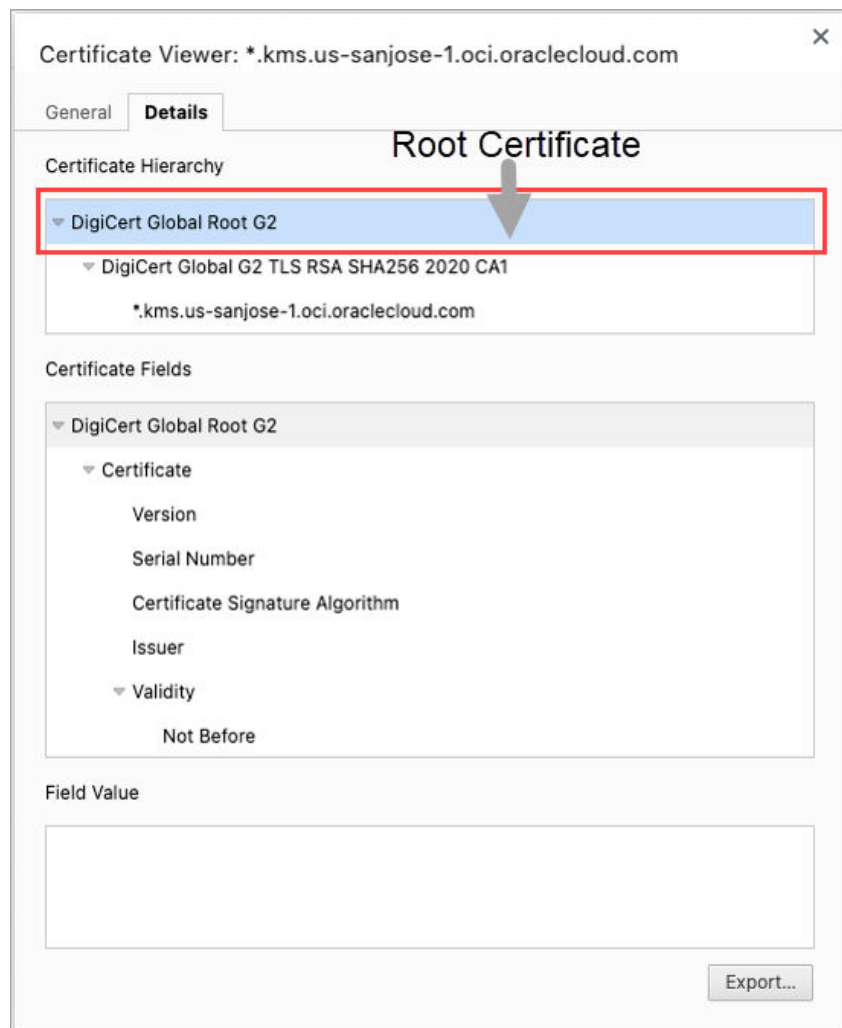
1. Navigate to **Identity & Security** page from the left-navigation pane and select **Vault** to open the Vault Information page.

2. From the Vault Information page, copy the cryptographic endpoint value and OCID.
3. Open a web browser and paste the cryptographic endpoint value in the browser URL bar. The browser does not display any page. However, you can click the **Connection is secure** to view the CA certificate.



This CA certificate is required by Oracle GoldenGate to be able to trust this OCI tenancy when connecting to it.

4. Go to the **Downloads** section of the web browser. The CA certificate are listed here.



5. Click **Export** to download the Root certificate.

✓ **Tip**

Keep the same directory for downloading the API key and the Root certificate.

Add the Digital CA Certificate as a Trusted CA Certificate in Oracle GoldenGate

The digital CA certificate which you downloaded previously using the cryptographic endpoint URL, needs to be added to the Oracle GoldenGate source deployment as a trusted CA certificate. See [Create Certificates for Secure Deployments](#).

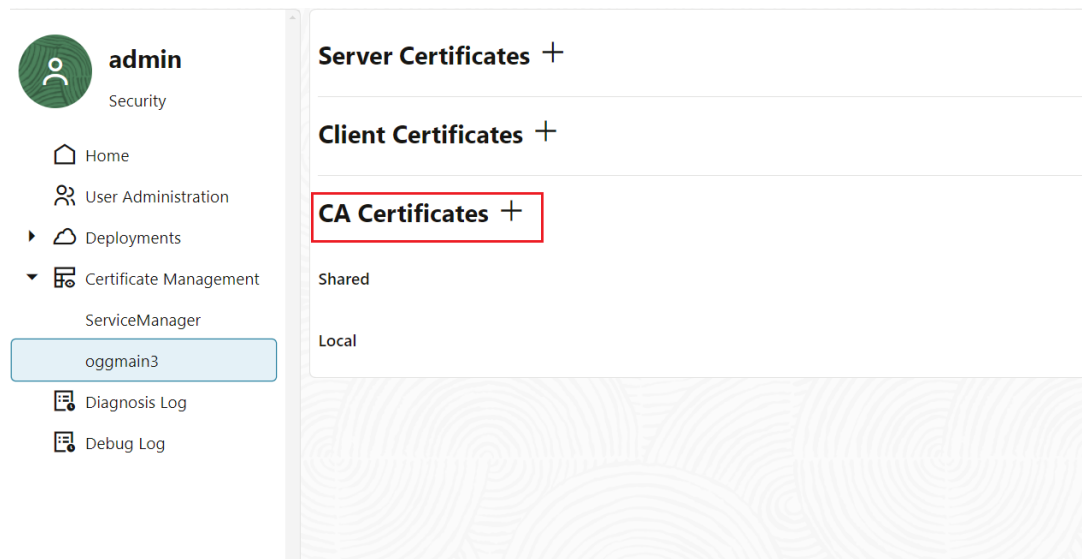
📘 **Note**

In OCI GoldenGate Service, you can skip this step as the CA certificate is already added as part of the service.

From the Oracle GoldenGate Service Manager, perform the following steps to add the digital CA certificate as a Trusted CA certificate for the source deployment:

1. Log in to Oracle GoldenGate Service Manager.

2. From the left-navigation pane, select the Certificate Management option.



Certificate Management page in Oracle GoldenGate Service Manager

As of now, there is no certificate added as a trusted certificate to connect to the specific OCI tenancy. You will need to add the root certificate that you had downloaded in step of

3. Add the root certificate as a trusted certificate to the CA Certificates in the Oracle GoldenGate deployment. This enables Oracle GoldenGate to trust a connection with the specific OCI tenancy. Also see, [Add a CA Certificate](#).

Add CA Certificate ✕

Add the CA Certificate

Unique Name

eastcert

Certificate PEM

```
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBCwggSjAgEAAoIBAQDjlGE4wV
UeTAggE
```



Certificate Location

☒ Shared ☐ Local

Submit

Configure OCI KMS to Connect with Oracle GoldenGate

From the OCI KMS tenancy, certain values are needed to set up a connection between Oracle GoldenGate and OCI KMS. These values are:

- Tenancy ID
- Cryptographic Endpoint
- User OCID
- API Key

Before configuring Oracle GoldenGate to connect with OCI KMS, you need to log in to the OCI tenancy to perform the following tasks:

- **Create a vault**, if not already created and get the **cryptographic endpoint** and **User OCID** values. See
- **Create and download an API private key pair** and information associated with the API key such as the **fingerprint** and **API key** value.
- **Download the CA certificate** using the **cryptographic endpoint**. This step is also a prerequisite for configuring Oracle GoldenGate encryption profile. See [Download the CA Certificate using the Cryptographic Endpoint](#) for details.

Use the following steps to view and save the OCI KMS values, which would be required while setting up Oracle GoldenGate processes for connecting to OCI KMS:

1. [Create a Vault](#).

2. [Generate the Master Key and Download the API Private Key.](#)

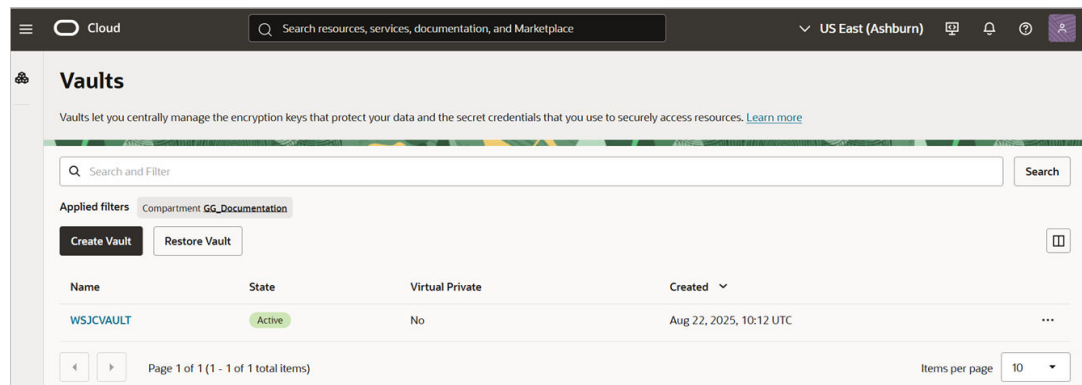
Create or Access the OCI Vault

Access the vault if it already exists in your OCI tenancy to determine the values for:

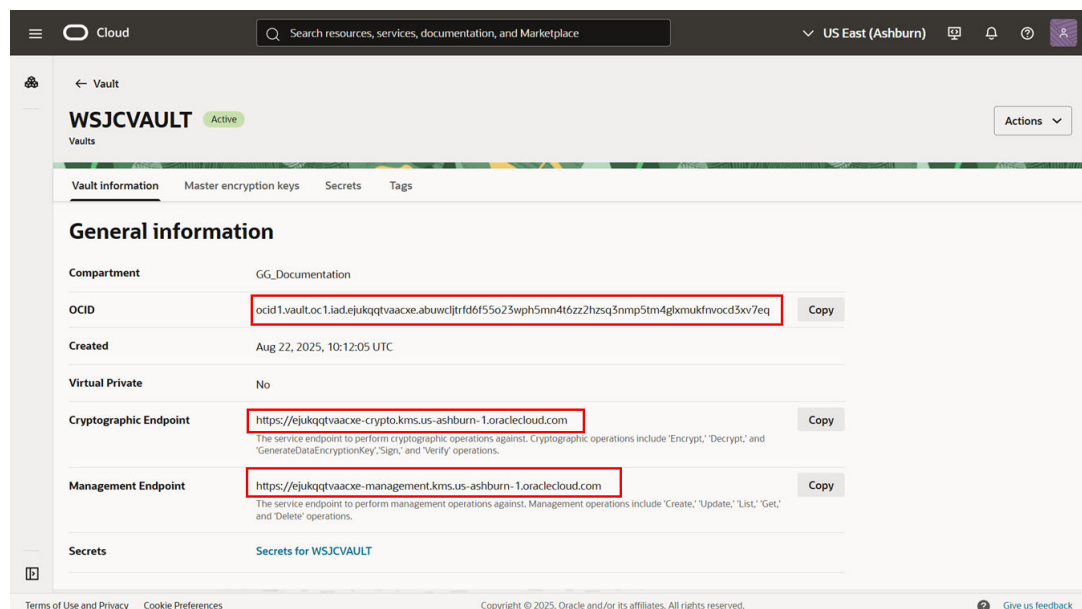
- **Cryptographic Endpoint:** This is the link from where you need to obtain the trusted certificate. Copy this link for use in the later steps.
- **OCID:** This is the unique ID for your OCI environment. It will be required while setting up the encryption profile in Oracle GoldenGate.

Use the following steps to create and access the vault:

1. Log in to your Oracle Cloud account.
2. From the left-navigation pane of the Oracle Cloud home page, click the **Identity & Security** option and then select **Vault**.
3. Click **Create Vault** to create a vault, if you haven't already created a vault. In this case, the vault (**WSJCVault**) is already created.



4. Click the vault name, for example **WSJCVault** shown in the following image, to access the information regarding vault ocid, cryptographic endpoint, and master key details. In the following image, the **General Information** section contains the **ocid** and **cryptographic endpoint** details and the **Master Encryption Keys in the Compartment_WH** section displays the master key details.



From this page, you get the values required to set up a trusted connection between Oracle GoldenGate and OCI KMS.

5. Copy this information to a notepad file for reference. In later steps, you will use this notepad file to add more details regarding the Master Key and API Key. These values would be used to add the encryption profile in the Oracle GoldenGate deployment.

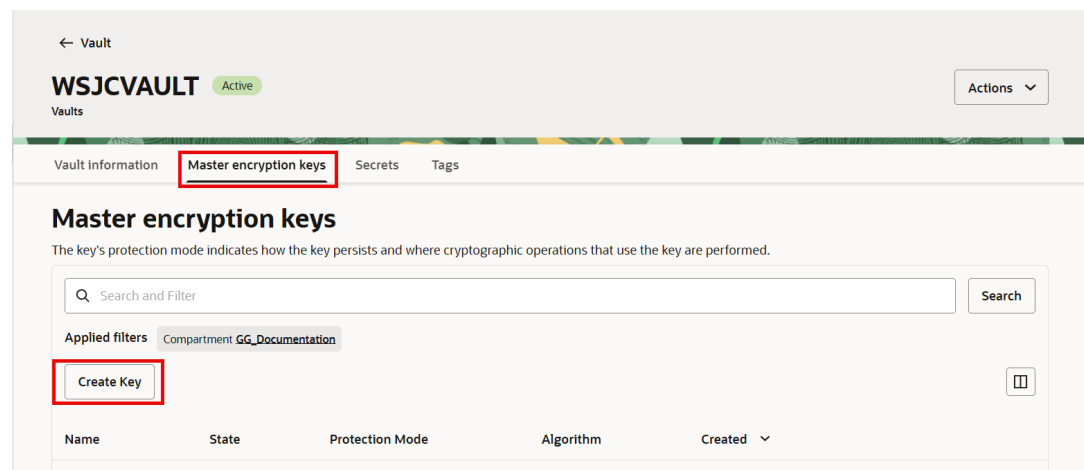
Generate the Master Key and Download the API Private Key

The following steps assume that you are already logged into your OCI tenancy.

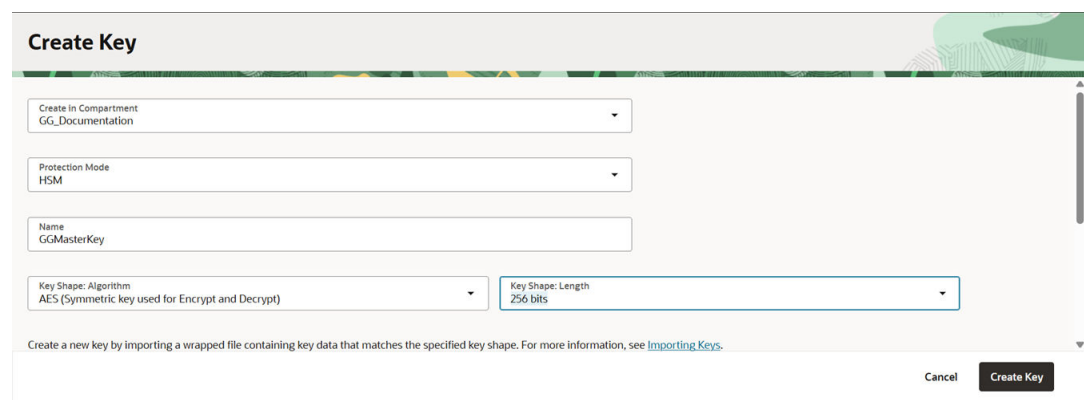
Generate Master Key

To create the master key:

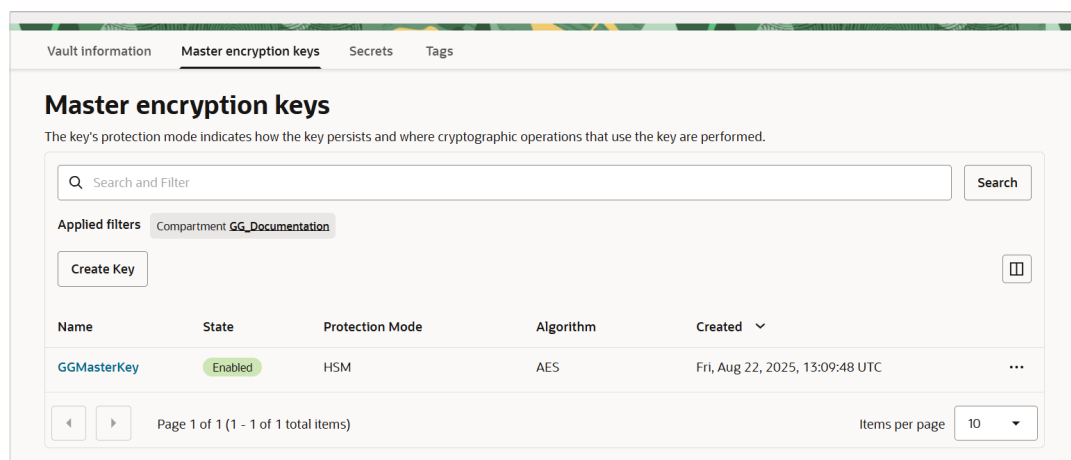
1. Navigate to the **Vault Information** page.
2. Move to the **Master encryption keys** tab and click **Create Key**.



3. Specify the name of the Master key and encryption algorithm among other details to create a master key, as shown in the following image.



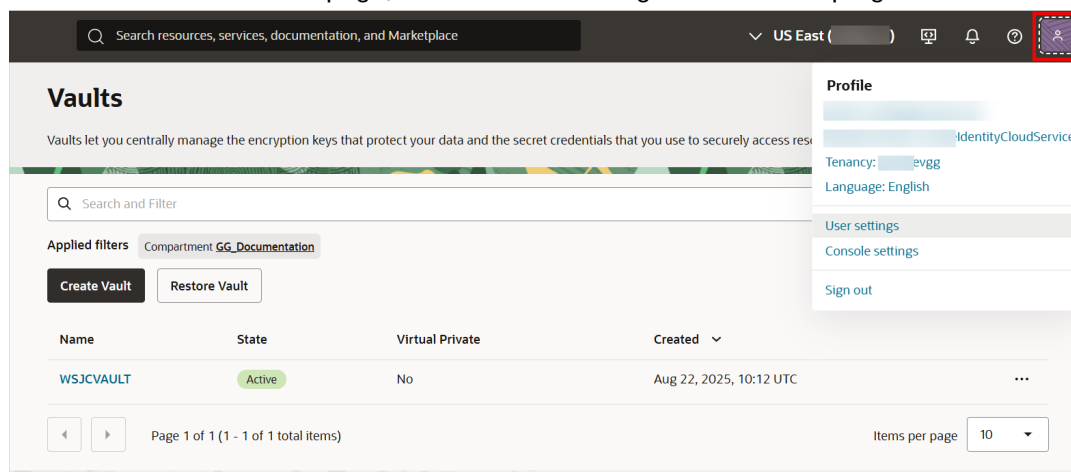
4. Click **Create Key**. This generates the master key that would be used by Oracle GoldenGate. The new Master Key is listed on the Master encryption keys page.



Generate API Key

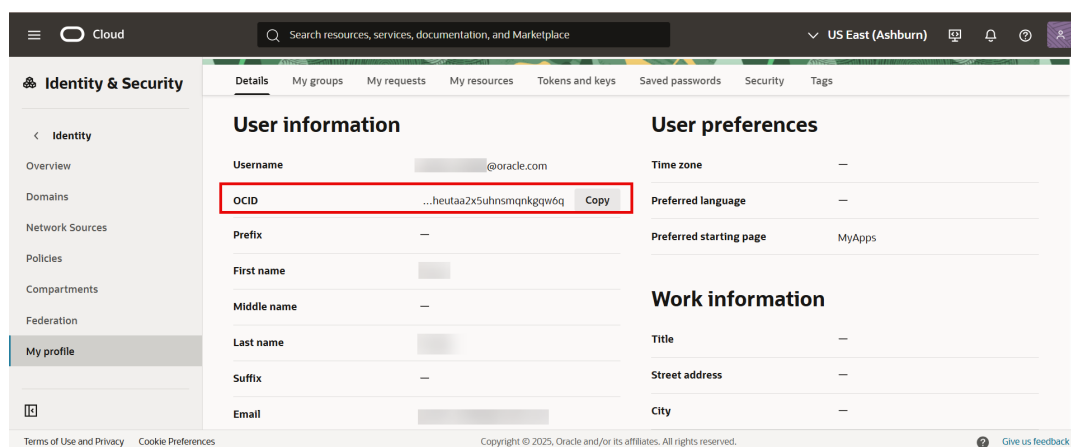
To connect the user with OCI KMS, create an API key using the following steps:

1. From the vault information page, click the User Settings icon on the top-right corner.

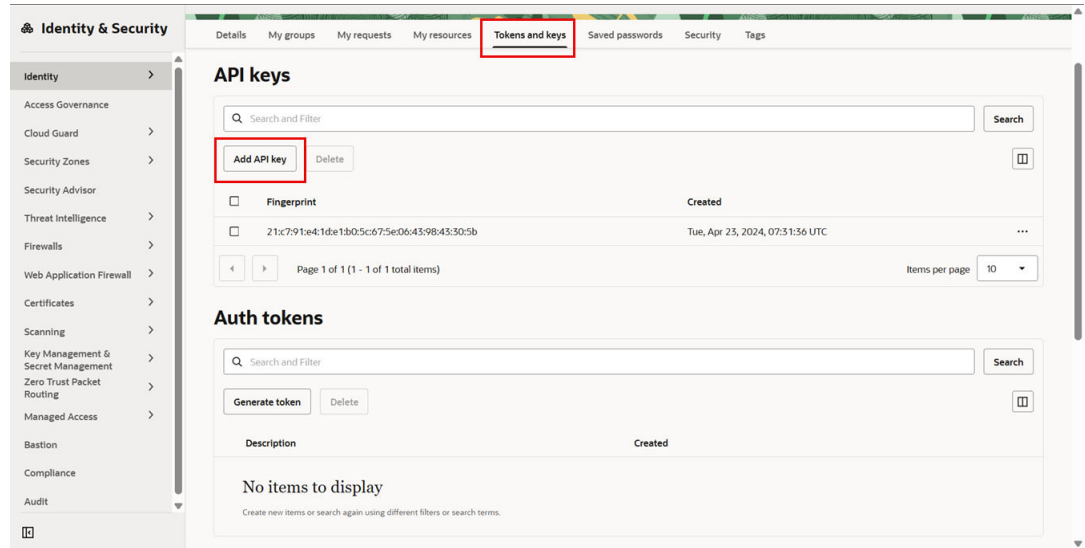


This displays the User Profile page with multiple tabs.

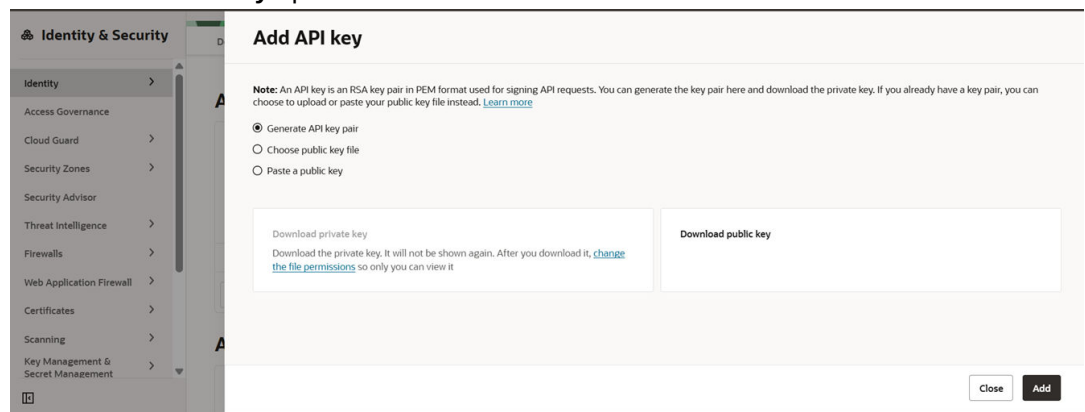
2. Copy the User OCID from the User Information section of the Details tab. Paste this value into the notepad file created earlier.



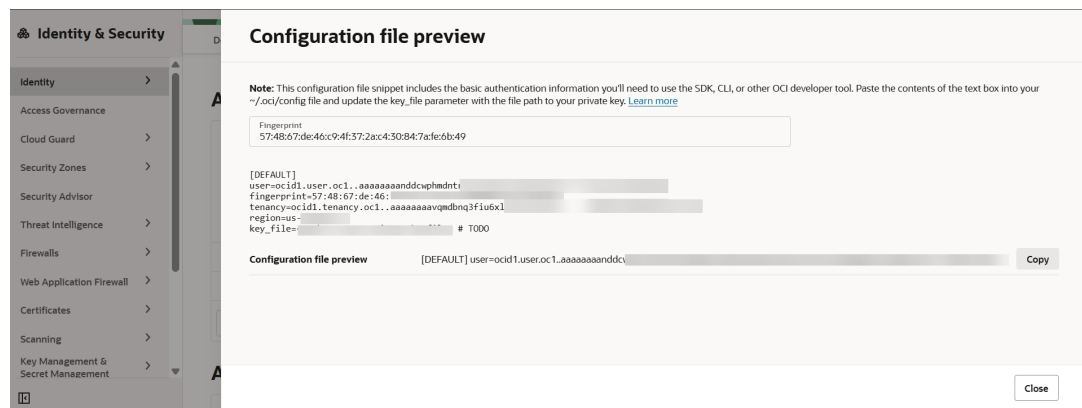
- Click the **Tokens and keys** tab to open the API Keys section.



- Click **Add API Key** to open the Add API Key dialog box.
- Select the **Generate the API key Pair** option to create a key pair for the OCI user to connect with OCI KMS. You also have the option to upload an existing public key file using the **Choose Public Key File** option or paste the value of public key in the text box using the **Paste Public Key** option.



- Click **Download Private Key** and keep it in a known location. You can rename the file to a user-friendly name such as **API_private_key.pem**.
- Click **Add**. This displays the **Configuration File Preview** dialog box, which contains all information associated with the API key such as the fingerprint, tenancy, region and other details.



- Copy and save these values in the same notepad file created earlier.

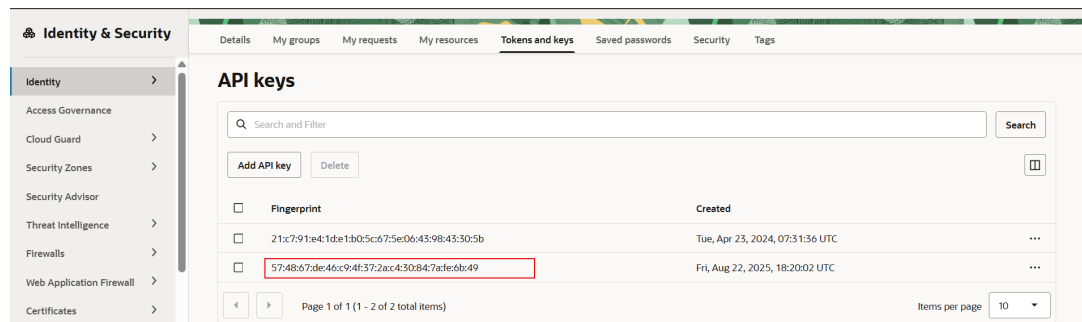
✓ **Tip**

Maintain the same notepad file to store information about the API key's fingerprint, tenancy value and the information about the cryptographic endpoint and OCID values for the OCI vault.

The following image shows a sample notepad file containing the configuration details for the vault, master key, and API key, that would be required while configuring the encryption profile in Oracle GoldenGate.



- Click **Close** to return to the API Keys section where the new API key is listed.



The next step is to set up Oracle GoldenGate encryption profile using all these details and then apply the encryption profile to Extract, Replicat processes, as needed. See the [Configure Oracle GoldenGate Processes to Enable OCI KMS Trail File Encryption](#) for next steps.

To learn about the OCI KMS encrypt and decrypt endpoints, see [/encrypt](#) and [/decrypt](#) endpoint documentation in *Oracle Cloud Infrastructure Documentation*.

Configure Oracle GoldenGate Processes to Enable OCI KMS Trail File Encryption

Before beginning the steps in this section, make sure that you have completed the [Prerequisites for Connecting Oracle GoldenGate with OCI KMS](#).

In the Oracle GoldenGate interface, perform the following tasks when configuring Oracle GoldenGate to set up a trusted connection with OCI KMS:

- Create an encryption profile using the OCID, cryptographic endpoint, API key, tenancy, and fingerprint values.
- Apply the encryption profile to Extract, Distribution Path, or Replicat processes.

Use the steps provided in the following sections to apply OCI KMS-based trail encryption from Oracle GoldenGate Microservices Architecture:

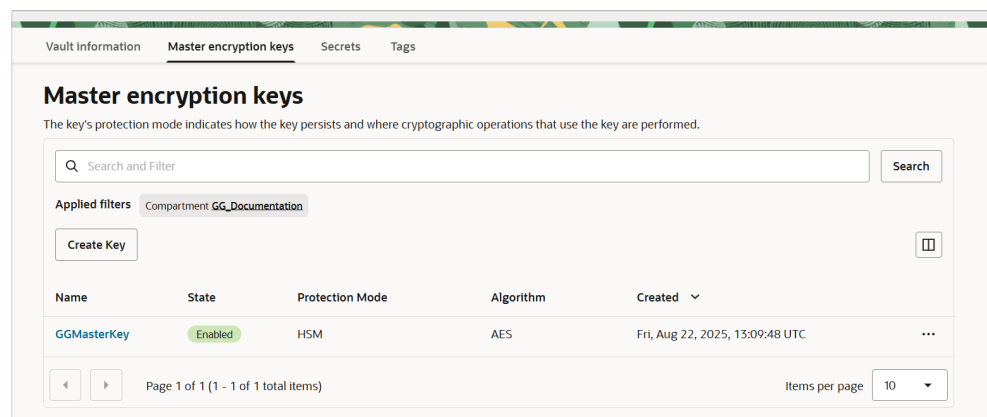
Create Encryption Profile in Oracle GoldenGate Processes

Use the following steps to apply OCI KMS-based trail file encryption from Oracle GoldenGate Microservices Architecture web interface:

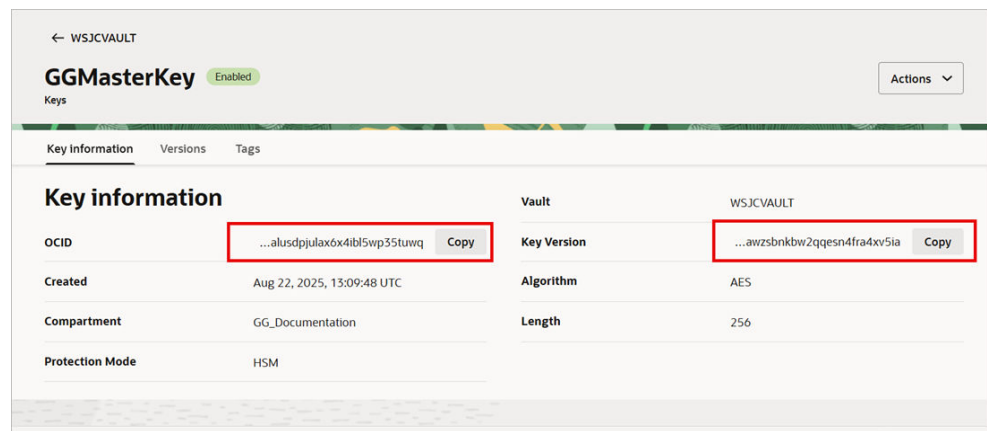
1. Open the notepad file where you saved the details for the OCI KMS API key and crypto endpoint details. See step 8 for reference from the [Configure OCI KMS to Connect with Oracle GoldenGate](#).

The information would include the following values:

- Crypto Endpoint URL: This value is displayed in the Vault page of OCI KMS.
- Tenancy OCID: This value can be obtained from the API values that were copied in the notepad file.
- Key OCID: To obtain this value:
 - a. Go to the OCI Vault page and click the master key to open the key details page where the OCID for the master key is provided.



- b. Copy the Master key OCID from the Key Details page.



- User OCID: Obtain this value from the API configuration details.
 - API Private Key: Upload the API Private Key from the location where you saved it while performing tasks in [Configure OCI KMS to Connect with Oracle GoldenGate](#).
 - API Key Fingerprint: Obtain this value from the API configuration details.
2. Log in to the Administration Service and select the **Encryption** and then select **Profiles** from the left navigation pane.
 3. Click the plus sign next to Oracle Cloud Infrastructure to open the Create Encryption Profile dialog box.

Create Encryption Profile ✕

Create an encryption profile for encrypting trails using a masterkey saved in OCI Key Management Service (KMS). Specify the crypto endpoint URL, tenancy OCID, key OCID, user OCID and the API key and fingerprint. Please also add the OCI KMS CA cert (via the ServiceManager) to the deployment truststore.

Profile Name
 OCIKMS

Description
 OCI KMS Encryption Profile

Encryption Profile Type
 Oracle Cloud Infrastructure (OCI)

Default Profile
☐

Crypto Endpoint URL
 https://ejukqqtvaacxe-crypto.kms:

Tenancy OCID
 ocid1.tenancy.oc1..aa

Key OCID
 ocid1.key.oc1.iad.

User OCID
 ocid1.user.oc1..aaaaaaaa

API Key
  
Enter 3000 or fewer characters.

Key Fingerprint
 57:48:67:de:46:

Submit

Add the values to the encryption profile and **Submit**. After you click Submit, the encryption profile is validated and if the validation is successful, then the OCI KMS encryption profile displays on the Encryption Profiles page, as shown in the following image:

Encryption Profiles

Oracle Key Vault + 🔍

Name ^	Default ▾	Description ▾	Path ▾	Oracle Key Vault Version ▾	Masterkey Name ▾	Time to Live: ▾	Actions
No data to display.							

Oracle Cloud Infrastructure + 🔍

Name ^	Default ▾	Description ▾	Crypto Endpoint URL ▾	Tenancy OCID ▾	Key OCID ▾	User OCID ▾	Actions
OCIKMS	No	OCI KMS Encryption Profile	https://ejukqqtvaacxe-crypto.OracleCloud.com	ocid1.tenancy.oc1..aaaaaafvnqrq region=us-ashburn-1	ocid1.key.oc1.iad.ejukp35tuwq	ocid1.user.oc1..aaagqw6q	 

4. You can change the values for the encryption profile by clicking the pencil icon, which opens the Edit Encryption Profile dialog box.

Apply the OCI KMS Encryption Profile for Extract

Use the following steps to implement the OCI KMS encryption profile for Extract:

1. From the Administration Service home page, click the plus sign next to Extracts to open the **Add Extract** dialog box.
2. After providing other details for the Extract, scroll down and expand the **Encryption Profile** section and select **OCIKMS** profile name.

Add Extract ×

1 ✓ Extract Information 2 ● Extract Options 3 ○ Downstream Capture 4 ○ Managed Options 5 ○ Parameter File

The Extract requires Trail Details and Source Credentials. Specify the required Extract details.

Source Credentials

Domain: domain1 Alias: alias1

Register to PDBs: CDB2_PDB1 X

Registration Options

CSN Optimized ☐

Extract Trail

Name: ea Subdirectory: Trail Sequence: 0 Trail Size (MB): 500

Encryption Profile: **OCIKMS** Encryption Algorithm: NONE

◀ Back Next ▶

3. In the Extract parameter file, include the ENCRYPTTRAIL AES256 option. The Extract parameter file would look similar to the following:

```
EXTRACT exte
USERIDALIAS ggwest DOMAIN OracleGoldenGate
ENCRYPTTRAIL AES256
EXTTRAIL ea
TABLE WPDB.U1.*;
```

4. Click **Create** to add Extract and then start the Extract.

5. On the target host, select **Add Replicat** from the Administration Service Overview page to add a Replicat.
6. Select the type of Replicat and populate the Replicat details.
7. Scroll to the **Encryption Profile** section, and select the same OCIKMS encryption profile (in this case OCIKMS). Click **Next**.
8. In the Replicat parameter file, include the DECRYPTTRAIL option. The Replicat parameter file looks similar to the following:

```
REPLICAT rencyt
USERIDALIAS ggeast DOMAIN OracleGoldenGate
DECRYPTTRAIL
MAP WPDB.U1.*, TARGET U2.*;
```

9. Create and then start the Replicat process.
10. If you want to apply the encryption profile on a Distribution Path (DISTPATH), then you need to do the following steps:
 - a. Create the OCI KMS encryption profile on the target host.
 - b. Create the DISTPATH and apply the OCI KMS encryption profile to it. See [Add a Distribution Path](#).
 - c. Use the same encryption profile to decrypt the trail on the target. This implies that you use the encryption profile created on the target host, while adding a Replicat.

The next section describes steps to test that the committed transactions are captured and applied when using an encryption profile

See ADD ENCRYPTIONPROFILE, ALTER ENCRYPTIONPROFILE if you want to set up the encryption profile using the Admin Client.

Test Data Replication with Trail File Encryption Using OCI KMS

Test the trail file encryption on the source side and trail file decryption on the target side using the steps given in this topic.

Test Trail File Encryption in the Source Deployment

In [Configure Oracle GoldenGate Processes to Enable OCI KMS Trail File Encryption](#), the Extract is set up with the OCI KMS encryption profile.

In this example, you will be able to confirm that the encryption profile is being used by Extract by viewing the Extract report file.

To check if Extract is using the OCI KMS encryption profile:

1. From the Administration Service, click **Extract** and then select the Extract where the encryption profile is applied. You will be able to see the OCI KMS encryption profile listed under the Encryption section of the Extract page.
2. For troubleshooting purposes, you can check if the trail file is encrypted at source, using Logdump commands. See the Scanning a Trail File to Check for Trail File Encryption from the *Logdump Reference for Oracle GoldenGate*.

Test the Trail File Decryption on the Target Deployment

On the target side, the following example tests that Replicat applies the 3000 transactions that were captured from source. To make sure that the Replicat is using the OCI KMS encryption profile to decrypt the trail file, check the Replicat report file.

1. From the Administration Service home page, click **Replicat**, and select the Replicat name.
2. Click **Statistics**. On the Statistics page, the applied transactions are displayed.
3. Check the Replicat main page to see if the encryption profile is implemented and used by the Replicat.

✓ Tip

To check if the trail data is received in encrypted format on the target, you can run Replicat without the `DECRYPTTRAIL` parameter. In this case, the Replicat report file displays that the trail data is encrypted and could not be decrypted without proper key setting.

With these use cases, you can test that the trail file on the Extract side is using OCI KMS encryption profile to encrypt the trail data. On the Replicat side, the OCI KMS encryption profile is used to decrypt the trail data and apply the transactions on the target.

Managing Identities in a Credential Store

Oracle GoldenGate uses credential stores to maintain encrypted database passwords and user IDs and associate them with an alias.

It is the alias, not the actual user ID or password, that is specified in a command or parameter file. No user input of an encryption key is required within the Oracle Credential Store Framework (CSF) embedded into Oracle GoldenGate.

A credential store can be used across multiple deployments with the same Service Manager user access, while retaining control over their local credentials.

You can partition the credential store into logical containers known as **domains**, for example, one domain per installation of Oracle GoldenGate. Domains enable you to develop one set of aliases and then assign different local credentials to those aliases in each domain. For example, credentials for user `ogg1` can be stored as `ALIAS ext` under `DOMAIN system1`, while credentials for user `ogg2` can be stored as `ALIAS ext` under `DOMAIN system2`.

Configure Credential Store from the Admin Client

1. From the Oracle GoldenGate installation directory, start the command line.
2. After using the `CONNECT` command to login to the deployment (when using the Admin Client), issue the following commands to perform various tasks with the credential store.

Command	Description
<code>ADD CREDENTIALSTORE</code>	Adds a database credential store.
<code>ALTER CREDENTIALSTORE</code>	Adds each set of credentials to the credential store.

Command	Description
INFO CREDENTIALSTORE	Retrieves information about an Oracle GoldenGate credential store. This information includes the aliases that a credential store contains and the user IDs that correspond to them. The encrypted passwords in the credential store are not returned.
DELETE CREDENTIALSTORE	Removes a credential store from the system. The credential store wallet and its contents are permanently deleted.

You can also perform these tasks from the Oracle GoldenGate web interface. See [Add Database Connections](#). To use the REST API service endpoint for performing this task, see [Credentials REST Endpoints](#)

Specify the Alias in a Parameter File or Command

The following commands and parameters accept an alias as substitution for a login credential.

Table 4-2 Specifying Credential Aliases in Parameters and Commands

Purpose of the Credential	Parameter or Command to Use
Oracle GoldenGate database login.	USERIDALIAS <i>alias</i>
Oracle GoldenGate database login for a downstream Oracle mining database.	TRANLOGOPTIONS MININGUSERALIAS <i>alias</i>
Password substitution for {CREATE ALTER} USER <i>name</i> IDENTIFIED BY <i>password</i> .	DDOPTIONS DEFAULTUSERPASSWORDALIAS <i>alias</i>
Oracle GoldenGate database login from the Admin Client.	DBLOGIN USERIDALIAS <i>alias</i>
Oracle GoldenGate database login to a downstream Oracle mining database from the Admin Client.	MININGDBLOGIN USERIDALIAS <i>alias</i>

Encrypt and Store User Credentials

As you set up and install Oracle GoldenGate, you must occasionally log-in to the database by using the DBLOGIN command, for tasks such as adding supplemental logging with the ADD TRANDATA command.

Encrypting the login password is a recommended security measure. However, using a secure password in the standard DBLOGIN command requires first encrypting it by using the ENCRYPT PASSWORD command. To avoid this step while protecting the user ID from exposure, you can create an Oracle GoldenGate credential store before you start setting up and configuring the user credentials.

When you use a credential store, you only have to supply an alias for the login credential whenever you log in with DBLOGIN. The credential store also makes the work of specifying login credentials for the Extract and Replicat processes easier and more secure when configuring the parameter files. You can create basic entries in the credential store at first and then use the management commands to expand it as needed. You can create an encryption profile using the Admin Client to set up your credential store.

Configure Kerberos Authentication

To enable Kerberos authentication in Oracle GoldenGate for Oracle database, the following configurations are assumed:

- Kerberos KDC is configured, and Kerberos system is installed locally.
- Kerberos Principals are configured for externally authenticated Database Users.
- Kerberos Caches are configured locally for each Kerberos Principal.
- Oracle Net Services are configured properly.
- Oracle Server parameter files are configured with Kerberos related settings.
- Externally authenticated database users are created with proper privileges.
- TNS_ADMIN environment variable is configured for Oracle GoldenGate.

Implement and Use a Kerberos Account to Access Oracle Database from Oracle GoldenGate

To begin the Kerberos authentication in Oracle GoldenGate Microservices Architecture, you need to first create a database user account alias prior to using DBLOGIN.

```
CONNECT http://localhost:9005 as ggadmin password We1come_$(
```

The following sample uses the deployment demo to set up user ID alias for a kerberos account:

```
ALTER CREDENTIALSTORE ADD USER  
/@EAST nopassword alias dbeast
```

```
2020-06-22T21:08:33Z INFO OGG-15102 Credential store created.  
2020-06-22T21:08:33Z INFO OGG-15114 Credential store altered.
```

```
INFO CREDENTIALSTORE
```

```
Default domain: OracleGoldenGate  
Alias: dbeast  
Userid: /@EAST
```

```
DBLOGIN USERIDALIAS dbeast
```

Successfully logged into database EAST.

Here, the NET SERVICE is the simple name for the database service. Alternatively, a complete connect string (descriptor) can be used instead of the Oracle net service name.

Here's an example of a predefined net service name and connect descriptor mapping:

```
EAST = (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=db1))  
(CONNECT_DATA=(SERVICE_NAME=EAST.regress.rdbms.test.us.oracle.com)))
```

A valid DBLOGIN command without USERID and password can then be specified as:

```
DBLOGIN USERID /@EAST
```

To know more, see the ALTER CREDENTIALSTORE, DBLOGIN USERIDALIAS, and MININGDBLOGIN commands. Also see USERIDALIAS parameters.

On the Oracle GoldenGate side, if you want to issue the DBLOGIN command with different externally authenticated users, the usage of a default Kerberos cache location is specified in the SQLNET.ORA file. This is then assumed to be the externally authenticated user for logging in to the database.

For example, observe a Kerberos Cache location specified in the client side SQLNET.ORA file:

```
SQLNET.KERBEROS5_CONF = /test/b/1234567890/oracle/work/krb/krb.conf
SQLNET.KERBEROS5_KEYTAB = /test/b/9876543210/oracle/work/krb/v5srvtab
SQLNET.KERBEROS5_CC_NAME = /test/b/1234506789/oracle/work/krb/krb.cc
```

In this example, the krb.cc is the Kerberos Cache used in this Oracle GoldenGate deployment. If you open the krb.cc cache file with the oklist utility, you can see that the default principal is used as the externally authenticated user oratst@US.ORACLE.COM.

```
[ demo_vw2 ] [demo@test02swv krb]$ oklist krb.cc
```

```
Kerberos Utilities for Linux: Version 21.0.0.0.0 - Production on 27-JUN-2025
23:59:13
```

```
Copyright (c) 1996, 2025 Oracle. All rights reserved.
```

```
Configuration file : /test/b/1234567890/oracle/work/krb/krb.conf
Ticket cache: FILE:krb.cc
Default principal: oratst@US.ORACLE.COM
```

```
Valid starting      Expires            Service principal
06/27/20 12:12:34  06/28/20 12:12:34  krbtgt/US.ORACLE.COM@US.ORACLE.COM
06/27/20 12:12:34  06/28/20 12:12:34  oratst/
demo2swv.us.oracle.com@US.ORACLE.COM
```

Configure Kerberos Authentication with MA

Here are the steps to configure kerberos authentication from the Admin Client.

Connect to the Administration Service from the Admin Client:

```
CONNECT http://localhost:9005 DEPLOYMENT oggdep as ggadmin PASSWORD We1come_$
```

Alter the credentialstore after connecting to the Administration Service of the deployment oggdep:

```
ALTER CREDENTIALSTORE ADD USER /@DBEAST NOPASSWORD ALIAS ggeast
```

Output shows:

```
2020-06-22T21:08:33Z INFO OGG-15102 Credential store created.
2020-06-22T21:08:33Z INFO OGG-15114 Credential store altered.
```


Run the following command to verify that the credentialstore was altered successfully:

```
INFO CREDENTIALSTORE
```

Output displays the following:

```
Default domain: OracleGoldenGate
Alias: ggeast
Userid: /@DBEAST
```

When using the MA web UI to create the credential, if the **User ID** field begins with a **/** character, then the password is not required. So, in the **User ID** field, enter */connect_string* where *connect_string* is your connection string.

Here, the NET SERVICE is the simple name for the database service. Alternatively, a complete connect string (descriptor) can be used instead of the Oracle net service name.

Here's an example of a predefined net service name and connect descriptor mapping:

```
DBEAST = (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=db1))
(CONNECT_DATA=(SERVICE_NAME=DBEAST.regress.rdbms.test.example.com)))
```

Example: Using USERIDALIAS in Parameter File for Kerberos Account

The following example shows how to set the USERIDALIAS values in the parameter file after creating the credential store with Kerberos authentication:

```
ALTER CREDENTIALSTORE ADD USER /@ggadmin NOPASSWORD ALIAS ggadmin
```

```
2020-12-17T21:08:33
INFO    OGG-15102 Credential store created.2020-12-17T21:08:33
INFO    OGG-15114 Credential store altered.
```

```
ALTER CREDENTIALSTORE ADD USER /@ggadmin_mining NOPASSWORD ALIAS
ggadmin_mining
```

```
2020-12-17T21:09:45
INFO    OGG-15102 Credential store created.2020-12-17T21:09:45
INFO    OGG-15114 Credential store altered.
```

```
INFO CREDENTIALSTORE
```

```
Default domain: OracleGoldenGate
Alias: ggadmin
Userid: /@ggadmin
```

```
Default domain: OracleGoldenGate
```

```
Alias: ggadmin_mining  
Userid: /@ggadmin_mining
```

After altering the credentialstore, you can specify USERIDALIAS options in the parameter file:

```
USERIDALIAS ggadmin  
DOMAIN OracleGoldenGate  
TRANLOGOPTIONS MINUSERIDLIAS ggadmin_mining  
DOMAIN OracleGoldenGate
```

5

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.

Before Adding a Deployment

If you need a secure deployment (recommended for production databases, or unsecured networks) make sure to check the **Enable Security** option in the [Select Service Manager Options](#) screen of the OGGCA wizard.

Start the OGGCA Wizard

Adding deployments is the first task in the process of setting up a data replication environment. Deployments are managed from the Service Manager.

After completing the Oracle GoldenGate Microservices Architecture installation, you can add initial and subsequent deployments using the Oracle GoldenGate Configuration Assistant (OGGCA) wizard.

You can also run OGGCA in silent mode. For steps to run OGGCA in silent mode, see [Add a Deployment in Silent Mode using OGGCA](#).

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. On Linux, run the oggca.sh program.

or,

On Windows, right-click the oggca.bat program.

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 wizard 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 port and other settings as configured for the existing Service Manager. The deployment would be added to this Service Manager.

Service Manager Deployment

ORACLE | GoldenGate

Service Manager Deployment

☐ Existing Service Manager ☒ New Service Manager

Software Home

Deployment Home

Server Settings

Hostname/IP Address Port

☐ Register as a service/system daemon ☐ Integrate with XAG

Remote Metrics for the Service Manager

☐ StatsD Host Port

Security

☐ Enable Security Server Certificate

☐ TLS 1.2 Server Private Key

☒ TLS 1.3 Server CA Certificate

☐ Enable FIPS

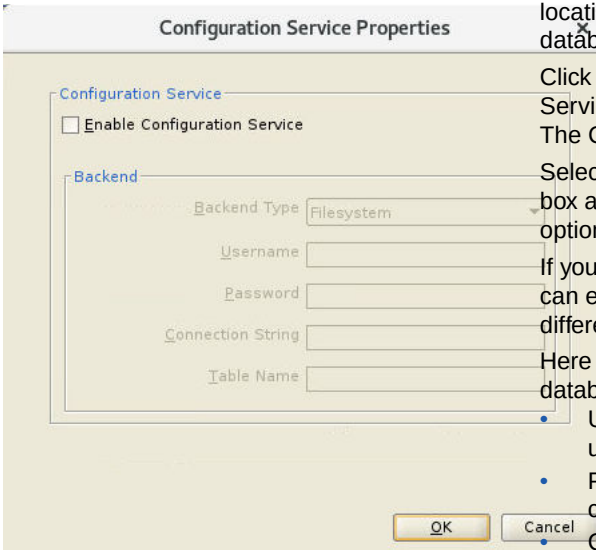
☒ Enable Plugin Service

2. For new Service Manager deployment, configure the options as described in the following table:

Option	Description
Create New Service Manager	This option is preselected when you start OGGCA for the first time, and no other Service Manager instance is running.
Software Home	Browse and select the Oracle GoldenGate software location. This is the OGG_HOME environment variable.
Deployment Home	For a new Service Manager deployment, browse and enter the directory that you want to use for storing the deployment directories. Oracle recommends that you create a ServiceManager directory within the deployment sub-directory structure to store the Service Manager files. The deployment home should never be a subdirectory of the Software Home (or OGG_HOME).

Option	Description
Hostname/IP address	Enter the server details for the Oracle GoldenGate instance. This field allows you to provide the values in different formats, as required. The available formats for adding hostname/IP address are: • Hostname/IP Address is 127.0.0.1: This address format refers to the local machine and is commonly used to represent the localhost. This option is ideal for testing or development environments where all services are hosted locally on the machine. With this address, you can connect to all microservices that are running on the local machine. Connections will work using both options, <code>127.0.0.1</code> and <code>localhost</code> in the URL. • Hostname/IP Address as the actual IP address: This address format allows you to connect to the microservices using <code>localhost</code>, <code>127.0.0.1</code>, or as the actual IP address. This is useful in scenarios where the system needs to be accessed both locally and externally, for example, from other machines in the same network. • Hostname as a Properly Resolved Hostname: This address format uses a resolvable hostname to, which is associated with a corresponding IP address, allowing connections through the resolved IP. You can connect to the services using <code>localhost</code>, the selected hostname, or the actual IP address associated with the hostname. This option is recommended for environments where the machine is part of a larger network and its hostname is properly configured for name resolution.
Listening Port	Enter the port number where Service Manager would be listening to service calls.
Register Service Manager as a system service/daemon	Select this check box to avoid manually starting and stopping it if the machine is rebooted. This ensure that Service Manager automatically starts with a server boot or restarts after a reboot of the server.
Integrate with XAG	Select this option to integrate your deployment with an Oracle Grid Infrastructure for Oracle. This option is only applicable when using clustered environments managed by Oracle Grid Infrastructure.
StatsD	Enable the StatsD check box to enable the recording the metrics for the Service Manager. Also see Configuration Service to know more about considerations for StatsD usage.
StatsD Host	Enter the hostname or IP address of the StatsD server. This information would be used by the Service Manager to connect to the StatsD host.

Option	Description
StatsD Port	Enter the port number for the StatsD host. The default port number is 8125.
Enable Security	Selected by default. Specify the security options to be configured with the Service Manager. Using the security options, you can add server, client, and trusted rootCA certificates and also set the protocol options for TLS. There are various ways to set up security in a deployment. See Secure Deployments to know more. To set up server, client certificates, see Manage Certificates for Deployments.
TLS 1.2	Select this check box to use the TLS 1.2 protocol.
TLS 1.3	Select this check box to use the TLS 1.3 protocol.
Server Certificate	Browse and select the server certificate file (.pem) that you want to associate with the Service Manager. See Create RootCA and Server Certificates to know about generating these type of certificates.
Server Private Key	Browse and select the server certificate private key file (.key) for the server certificate.
Server CA Certificate	Browse and select the trusted RootCA certificate for the Service Manager. If the Server Certificate is signed by one or more intermediate certificates, use the combined intermediate certificates and root certificate as the Server CA Certificate. The certificate location is defined by the OGG_CLIENT_TLS_CAPATH environment variable.
Plugin Service	Select this check box to enable the Plugin Service, which allows using third-party software with Oracle GoldenGate such as the database password plugin service.
Enable FIPS	Enable Oracle GoldenGate services to be compliant with the Federal Information Processing Standards (FIPS).

Option	Description
Configuration Service 	Enable this service to store the Oracle GoldenGate configuration files on a different location such as a high available filesystem or database. Click the Configuration Service button on the Service Manager Deployment Options screen. The Configuration Service dialog appears. Select the Enable Configuration Service check box and set up the back end options. The default option is Filesystem. If you choose the Oracle Database option, you can either use the source/target database or a different database outside the replication. Here are the details you need to specify for the database: • Username: The username of the database user. • Password: Password for the database user credential. • Connection String: The URL used to connect to the database. This is the backend connection string. The connection string can use the data source in the format: host[:port]/service_name or a TNS_ALIAS. For example: <i>localhost:1521/cdb1_pdb1.rdbms.oracle.com</i> is an example of hostname, port number, and service name format. • Table Name: Database backend table name where the configuration files would be stored. The backend table might be in any schema. In this example, the Oracle GoldenGate Admin schema (ggadmin) is used. Example: ggadmin.ggs_backendtable. For more details on using the Configuration Service, see Configuration Service.

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.

User Deployment

Configure the deployment options from the User Deployment screen, as described in the following steps:

1. Use the following table to configure the deployment options.

Option	Description
Deployment Name	Specify a deployment name.
Deployment Home	Browse and select the location of your deployment. The deployment home should never be a subdirectory of the Software Home (or OGG_HOME).
Customize	Click the Customize button to specify a different location for the configuration directories: OGG_ETC_HOME: Deployment etc home. OGG_CONF_HOME: Deployment Configuration Home. OGG_SSL_HOME: Deployment Security Home. OGG_VAR_HOME: Deployment variable directory home. OGG_DATA_HOME: Deployment data home. OGG_ARCHIVE_HOME Performance Metrics DataStore Home.
Ports	Enter the port numbers to be used by each of the Microservices: Administration Service Distribution Service Receiver Service Performance Metrics Service.
Data Store Type	Select the type of data store you need for storing Performance Metrics data. Options include BDB, and LMDB.
Data Store Home	Browse and select a location for the data store home directory.
StatsD	Enable this check box to specify the connection details of the StatsD server, which needs to be connected to the deployment. You need to specify the host name and a unique port number of the StatsD server using the options that get enabled after you select the StatsD enable checkbox. Also see Configuration Service to know more about considerations for StatsD usage.
StatsD Host	Enter the hostname or IP address of the StatsD server. This information would be used by the Service Manager to connect to the StatsD host. The StatsD server can be located on the same host machine where Oracle GoldenGate deployment exists, or it could be a remote host.

Option	Description
StatsD Port	Enter the port number for the StatsD host. The default port number is 8125.
Server Certificates	This box is disabled as the server certificate file (.pem) is already selected while configuring the Service Manager options.
Server Private Key	The server certificate private key file (.key) is preselected.
Server CA Certificate	The trusted RootCA certificate is also preselected.
Client Certificates	Browse and select the client certificate file (.pem) if required. To know how to add certificates after the deployment has been created, see Manage Certificates for Deployments.
Client Private Key	Browse and select the client certificate private key file (.key) for the associated client certificate.
Client CA Certificate	Browse and select the trusted rootCA certificate that would verify the associated client certificate. To know more about how to generate certificates for distribution client (DIST_CLIENT) and general client certificates, see Create External Trusted RootCA and Distribution Client Certificates
TNS_ADMIN	(Oracle only) Enter the value for this environment variable based on the location of the tnsnames.ora and sqlnet.ora files. The TNS_ADMIN parameter specifies the directory path of Oracle Net Services in which files such as the tnsnames.ora and sqlnet.ora reside. The tnsnames.ora file is the configuration file that contains the net service names that are mapped to the connect descriptors of the database services.
Replication Schema	Set the GGSCHEMA parameter for deployment. For more information about this setting, see GGSCHEMA in <i>Parameters and Functions Reference for Oracle GoldenGate</i> .
Environment Variables	Click this button to open the Environment Variables dialog box. It includes the paths for the preset environment variables, \$OGG_HOME, and \$LD_LIBRARY_PATH. Some variables are fixed, others can be extended, and you can add additional variables. For Db2 z/OS, make sure to set up the IBMCLIDRIVER and LD_LIBRARY_PATH before you start creating the deployment. To set up these environment variables, see Set Environment Variables for Oracle GoldenGate for Db2 z/OS.

- Click Next to move to the **User Deployment Administrator** screen.

User Deployment Administrator

On this screen, you can create another local administrator user specifically for the deployment. This user can be the same the Service Manager administrator also.

If you select the **Same as Service Manager administrator credentials** option, then you do not need to specify a different set of login credentials for the user deployment administrator. You will be able to log in to the Administration Service and other microservices using the same login credentials that are used to log in to the Service Manager.

Deselect the **Same as Service Manager administrator credentials** option to create a new administrator user for the deployment, specifically. Specify the username and password for the deployment administrator and use the strong password policy to ensure security.

This user would be able to log in to the Administration Service and other microservices. However, the Service Manager administrator log in credentials would be required to log in to the Service Manager.

Summary

Summary

1. Review the detailed configuration settings of the deployment before you continue.
2. (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

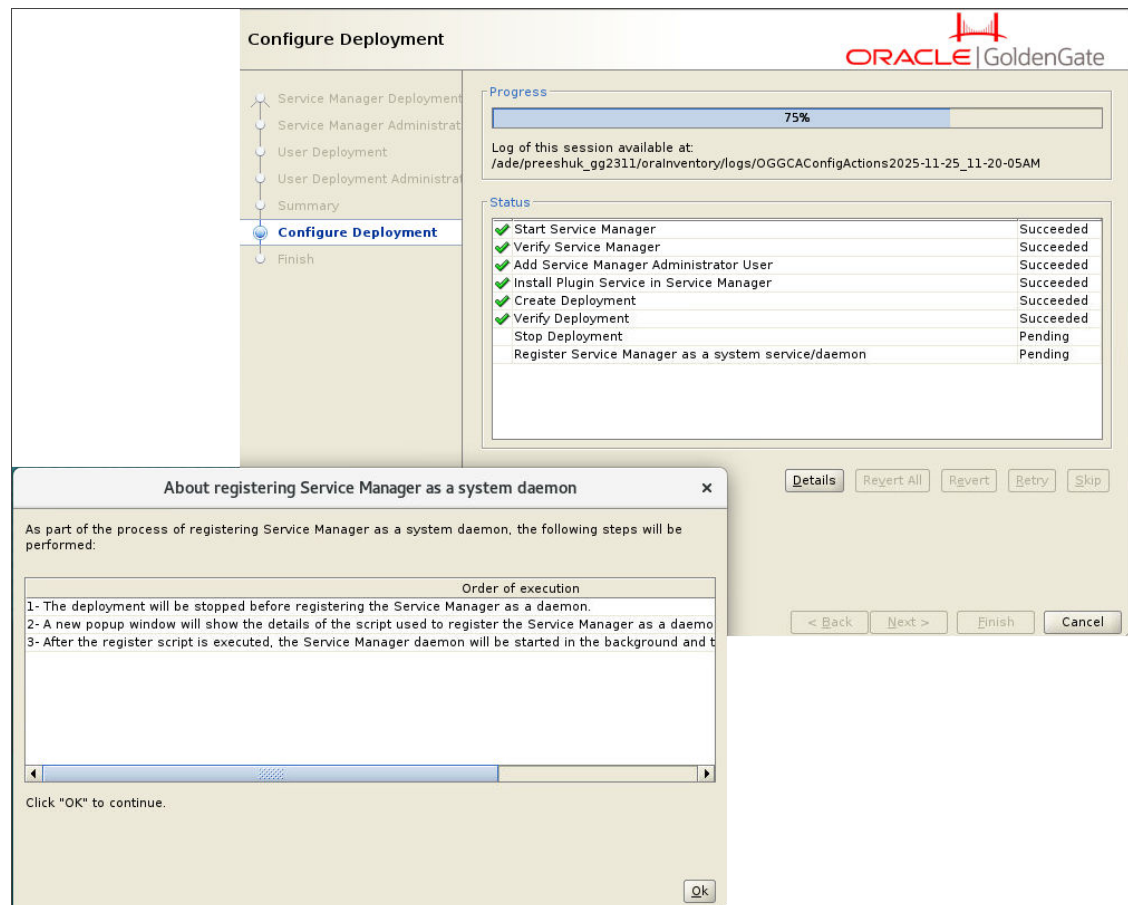
When saving to a response file, the administrator password is not saved for security reasons. You must edit the response file and enter the password if you want to reuse the response file for use on other systems.

3. Click **Finish** after reviewing the response file.
4. Click **Next**.

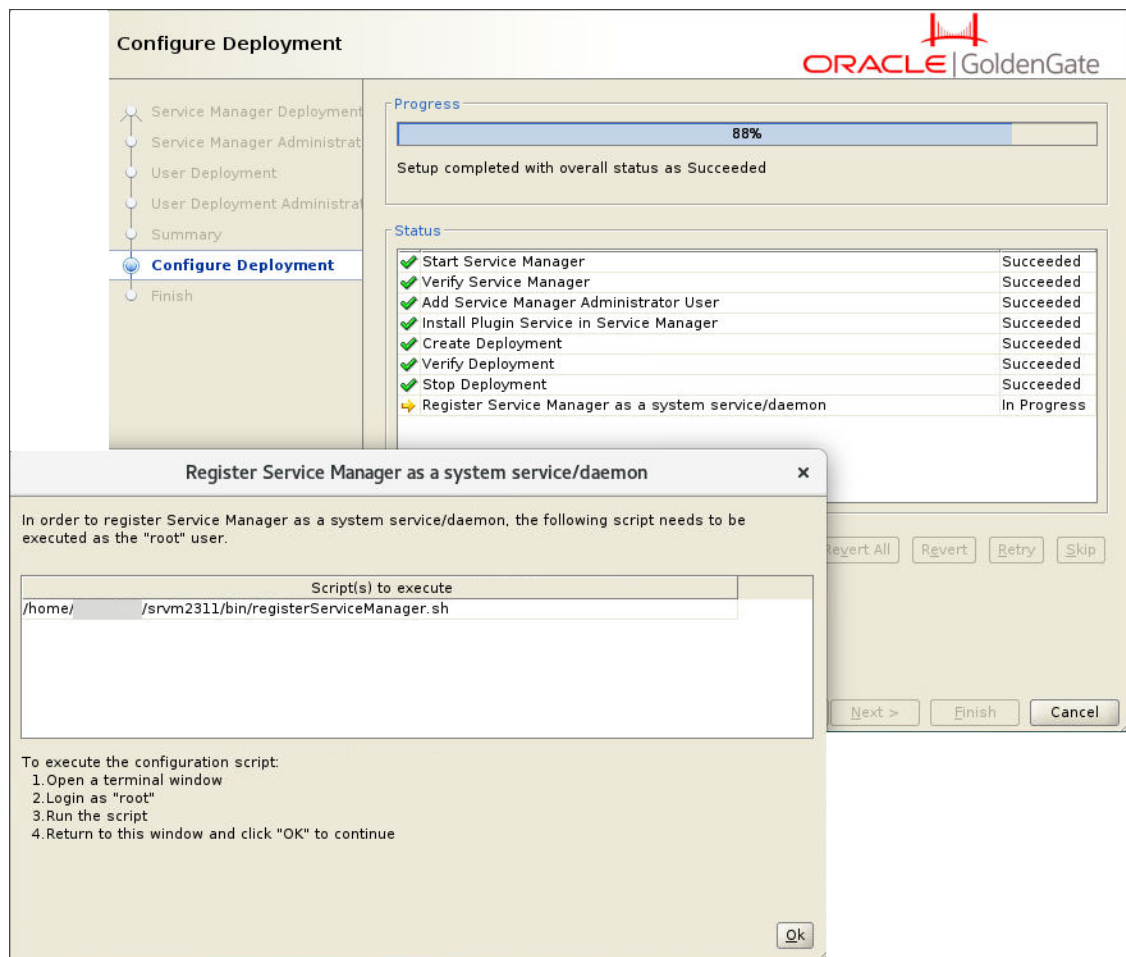
Configure Deployment

This screen displays the progress of the deployment creation and configuration.

If you had selected the **Register Service Manager as a service** check box in the Service Manager Deployment screen, then a pop-up box appears while the deployment is being created. Click **OK**.



At this stage the deployment is stopped and another pop-up box appears. Now, run the `registerservicemanager.sh` utility located in the `$SM_HOME/bin` directory as root user.



This script creates an inventory folder for the Service Manager. After the script run successfully, return to the **Register Service Manager as a system service/daemon** pop-up box and click **OK**. The OGGCA Configure Deployment screen displays that the deployment creation completed successfully.

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

Finish

Finish

Click **Close** to exit OGGCA.

Add a Deployment to an Existing Service Manager

Oracle recommends having a single Service Manager per Oracle GoldenGate hub. All deployments created on this hub, should be assigned to the single Service Manager on the host. However, this should not be done for XAG enabled Service Managers.

Note

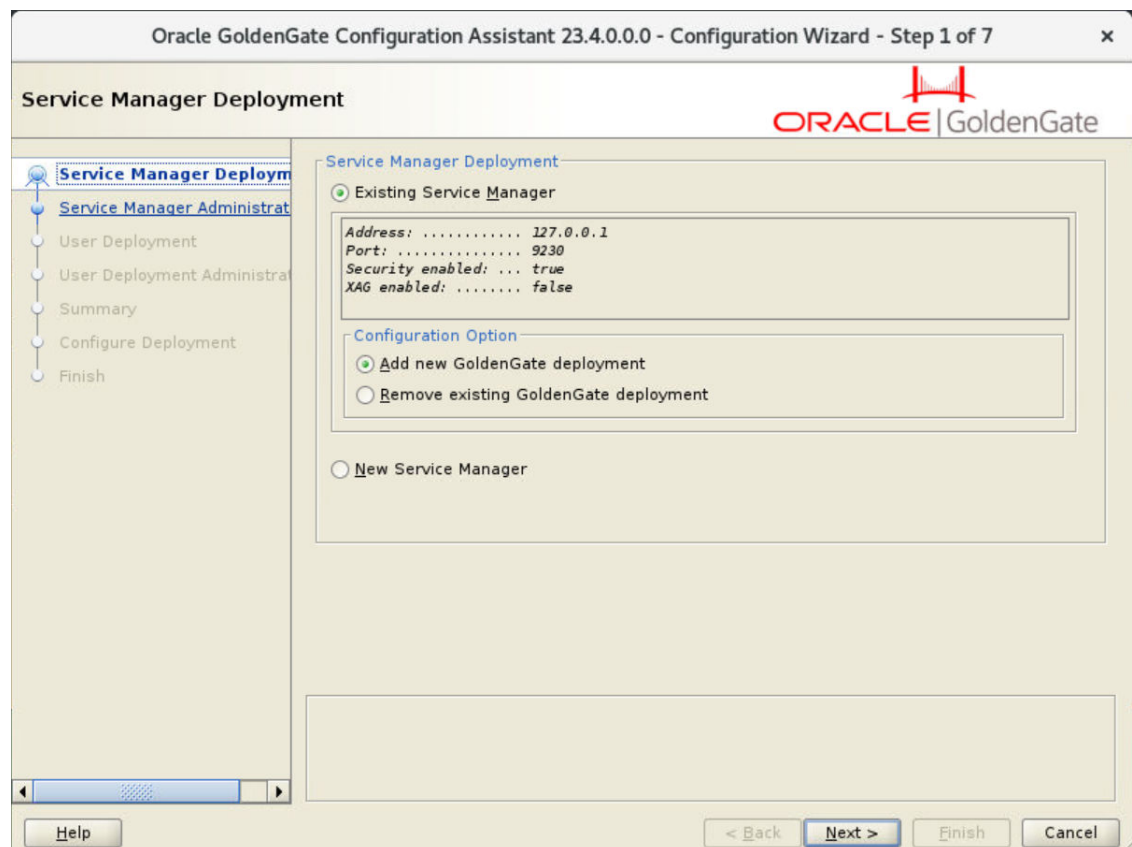
In case of a single Service Manager being used for multiple deployments for different databases, you must run the `oggca.sh` utility from the respective Oracle GoldenGate build for the specific database, when the deployments need to be created for database-specific Oracle GoldenGate build, which is different from the existing Oracle GoldenGate build.

For example, if you need to add a deployment for MySQL database to the existing Service Manager that is setup from PostgreSQL's Oracle GoldenGate build, then you need to run the `oggca.sh` utility from MySQL's Oracle GoldenGate build to perform the deployment.

To add a deployment to an existing Service Manager, run the OGGCA from the command line:

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

The OGGCA configuration wizard is displayed. **Existing Service Manager** option is preselected in the Service Manager Deployment screen.



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

1. From the Configuration Options section, select **Add new GoldenGate deployment** to add a deployment the existing Service Manager and click **Next**.

- On the Service Manager Administrator Account screen, specify the Service Manager login credentials for the existing Service Manager and click **Next**, as shown in the following image.

- On the User Deployment screen, configure the options for the new deployment, as mentioned in the following table:

Option	Description
Deployment Name	Specify a deployment name.
Deployment Home	Browse and select the location of you deployment. This location must be outside the Service Manager and Oracle GoldenGate installer locations.
Customize	Click the Customize button to specify a different location for the configuration directories such as OGG_ETC_HOME, OGG_CONF_HOME, OGG_SSL_HOME, OGG_VAR_HOME, OGG_DATA_HOME, OGG_ARCHIVE_HOME, and Performance Metrics DataStore Home.
Ports	Enter the port numbers for the Microservices: Administration Service Distribution Service Receiver Service Performance Metrics Service.

Option	Description
Data Store Type	Select the type of data store you need for storing Performance Metrics data. Options include BDB, and LMDB.
Data Store Home	Browse and select a location for the data store home directory.
StatsD	Enable this check box to specify the connection details of the StatsD server, which needs to be connected to the deployment. You need to specify the host name and a unique port number of the StatsD server using the options that get enabled after you select the StatsD enable checkbox.
StatsD Host	Enter the hostname or IP address of the StatsD server. This information would be used by the Service Manager to connect to the StatsD host. The StatsD server can be located on the same host machine where Oracle GoldenGate deployment exists, or it could be a remote host.
StatsD Port	Enter the port number for the StatsD host. The default port number is 8125.
Server Certificates	This box is disabled as the server certificate file (.pem) is already selected while configuring the Service Manager options.
Server Private Key	The server certificate private key file (.key) is preselected.
Server CA Certificate	The trusted RootCA certificate is also preselected.
Client Certificates	Browse and select the client certificate file (.pem) if required.
Client Private Key	Browse and select the client certificate private key file (.key) for the associated client certificate.
Client CA Certificate	Browse and select the trusted rootCA certificate that would verify the associated client certificate.
TNS_ADMIN	(Oracle only) Enter the value for this environment variable based on the location of the tnsnames.ora and sqlnet.ora files. The TNS_ADMIN parameter specifies the directory path of Oracle Net Services in which files such as the tnsnames.ora and sqlnet.ora reside. The tnsnames.ora file is the configuration file that contains the net service names that are mapped to the connect descriptors of the database services.
Replication Schema	Specify the schema name of the Oracle GoldenGate Database administrator user.

Option	Description
Environment Variables	Click this button to open the Environment Variables dialog box. It includes the paths for the preset environment variables, \$OGG_HOME, and \$LD_LIBRARY_PATH. Some variables are fixed, others can be extended, and you can add additional variables. For Db2 z/OS, make sure the set up the IBMCLIDRIVER and LD_LIBRARY_PATH before you start creating the deployment. To set up these environment variables, see Set Environment Variables for Oracle GoldenGate for Db2 z/OS. You may want to check the environment variable settings for the respective database from the databased-specific documentation.

4. On the User Administrator Account screen, enter the login credentials for the new deployment. Make sure to use a strong password. Click **Next**.
5. Review the configurations for the deployment on the Summary screen. Click **Finish** and then click **Next**.

(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.

6. View the progress of the deployment set up on the Configure Deployment screen. If the deployment is added successfully, then you'll see a message that the deployment is added successfully. Click **Next**.
7. Click **Close** on the Finish screen to exit OGGCA.

Add a Deployment in Silent Mode using OGGCA

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

1. Copy the sample response file provided at [OGGCA Sample Template Response File for Silent Deployment](#) and edit to complete the required Deployment configuration.
2. Save the response file with a .rsp extension, such as oggdeploy.rsp.

Note

Alternatively, if you have the ability to run OGGCA in interactive mode, you can follow the prompts and complete the required information, then save the steps as a response file, which is offered in the final step of the deployment creation. See [OGGCA Response File in Interactive Mode](#) for details.

3. Create the Deployment by running the following command:

Linux:

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

Windows:

```
ogg-home\bin\oggca.bat -silent -responseFile path\oggdeploy.rsp
```

Examples:

Linux:

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

Windows:

```
C:\ogg\bin\oggca.bat -silent -responseFile  
C:\ogg\bin\inventory\response\oggdeploy.rsp
```

Connect to the Deployment from the Web Interface

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

1. Open a web browser and enter the host URL where the Oracle GoldenGate Microservices deployment is installed.

The URL is similar to `http://host:port`, where `host` is the name or IP address of the server that is running the Service Manager and `port` is the port number of the Service Manager. For a secure deployment, the URL is similar to `https://host:9001`.

2. Enter the user name and password you created during deployment creation and sign in.

Start and Stop the Service Manager

The start and stop process of the Service Manager within Oracle GoldenGate Microservices Architecture is based on two approaches:

- If the Service Manager is configured in manual mode then there are scripts in `$OGG_SRVMGR_DIRECTORY/bin` that you can run to start or stop the Service Manager.
 - To start the Service Manager: `$OGG_SRVMGR_DIRECTORY/bin/startSM.sh`
 - To stop the Service Manager: `$OGG_SRVMGR_DIRECTORY/bin/stopSM.sh`

Note

Before running either script to start or stop the Service Manager, edit and set the following variables:

```
export $OGG_HOME=Oracle GoldenGate installation folder
```

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

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

- If the Service Manager is configured as a daemon or service, 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 Linux:

```
systemctl start OracleGoldenGate
```

```
systemctl status OracleGoldenGate
```

```
systemctl stop OracleGoldenGate
```

For Windows:

```
net start OracleGoldenGate
```

```
net stop OracleGoldenGate
```

- If you experience any issues while using the `registerServiceManager.sh/bat` utility to register the Service Manager as a daemon or service, see the MOS note 3051194.1 [RegisterServiceManager.sh Fails By oggca.sh: command not found](#). Also see the Known Issues and Limitations 23ai (23.9.1.25.08) in *Oracle GoldenGate Release Notes*.
- 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.

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:

```
$OGG_HOME/bin
```
2. Select **Existing Service Manager** from the **Select Service Manager Options** screen. Click **Next**.

3. If there are no other deployments to remove, then the option to remove the Service Manager is available in the drop down. Select **Remove Service Manager Deployment** from the Configuration Options screen.
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	• /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 *
```

① 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.

6

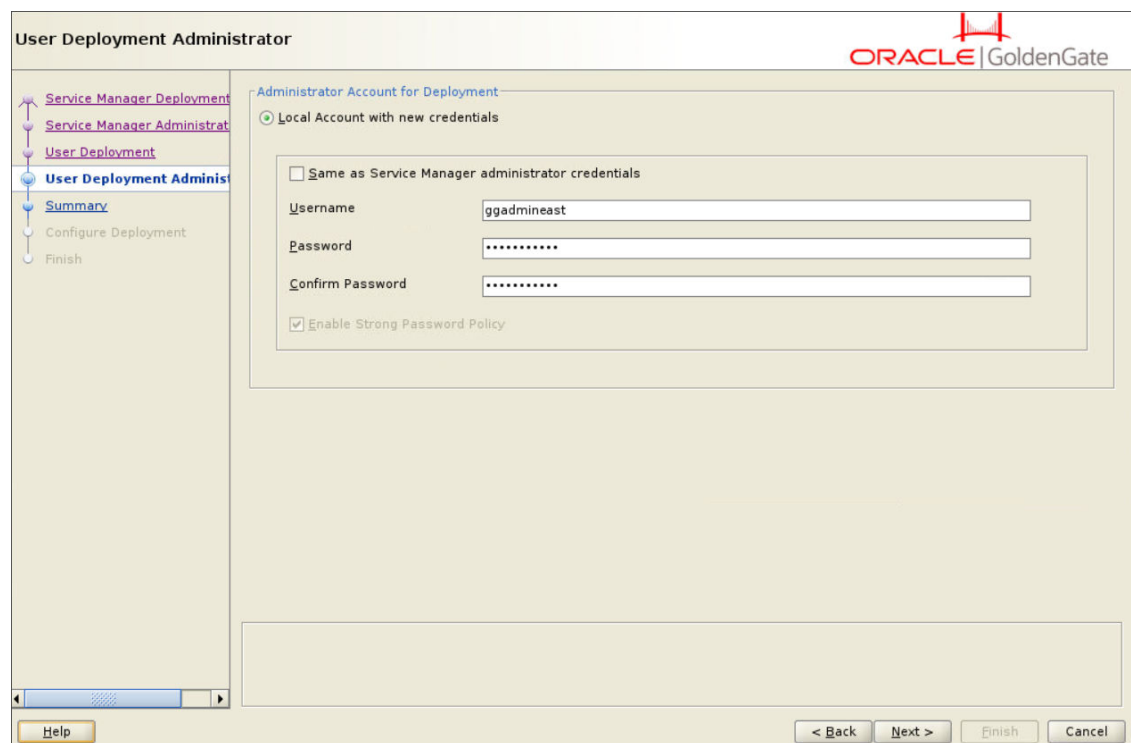
Manage Deployments

This section provides instructions to manage deployment, external identity provider authentication setups for a deployment, adding and editing user credentials among other key tasks performed after installing and deploying Oracle GoldenGate.

Add Users to a Deployment

Each deployment has its own set of users with specific roles. The Service Manager administrator user is created from OGGCA Service Manager Administrator screen. This user can connect to the Service Manager. This user does not have access to other microservices.

However, when you configure the administrator account for the microservices using the User Deployment Administrator screen in OGGCA, you can select to use the Service Manager administrator credentials to access the microservices as well.



As shown in the image, if you enable the "Same as Service Manager administrator credentials", then the other fields are disabled because the same user credentials are used to access the Service Manager and the deployment microservices.

Other users created from the Service Manager can only access the Service Manager. User created from the Administration Service web interface can access all microservices, but they cannot access the Service Manager.

These users are not available with other deployments on the same host server.

To create users from the Service Manager or Administration Service:

1. Log in to either the Service Manager or the Administration Service.
2. From the left navigation pane, select **User Administration**.
3. Click **Users** (+) to add users.
4. Enter a unique user name.
5. Select one of the roles from the **Role** list box. The options are **User**, **Operator**, **Administrator**, and **Security**.

Table 6-1 Oracle GoldenGate User Roles and Privileges

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 server.
Administrator	Grants full access to the user, including the ability to alter general, non-security related operational parameters and profiles of the server.
Security	Grants administration of security related objects and invoke security related service requests. This role has full privileges.

6. Select the user type from the **Type** list box as **Password** or **Certificate**.

If you select the user type as **Password**, then the authentication is done based on the username and password.

If you select the user type as **Certificate**, then the user will authenticate itself by presenting a client certificate. After you select the **Certificate** option, you need to enter the common name (in the certificate that will be presented such CN="certuser").

Note

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

7. Enter information that describes the user.
8. Click **Submit**. The user is registered.

Edit Users

User role cannot be changed. You must delete a user and add it, as required. However, you can modify or edit the following user attributes:

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

To edit user attributes:

1. Navigate to the **User Administration** page from the **Service Manager** or **Administration Service**.
2. Click the **Edit User** (pencil) in the **Action** column of the **Users** table.
3. Change the required attribute.
4. Click **Submit** to confirm the modifications to the user attributes.

Change the Administrator User Password Registered with XAG for a Deployment

To change the password for the Oracle GoldenGate Microservices administrator account that is registered with XAG, through the web interface or the REST API. The following steps guide you to change the password from the REST API:

1. Disable the XAG healthcheck monitoring, which is run by Grid Control, preventing it from failing and stopping GoldenGate processes during the password changing process:

```
$ agctl disable goldengate instance name
```

2. Change the Service Manager and Deployment administrator password:

```
curl -u USER:PASSWORD -X PATCH http://hostname.us.com:7701/services/v2/
authorizations/security/oggadmin
-H "Content-Type:application/json"
-H "Accept:application/json" -d
@update_security_password.json| python -mjson.tool
```

In this code syntax, replace the hostname and port number with the deployment server hostname and port number for the deployment.

The contents of the `update_security_password.json` script are as follows:

```
$cat update_security_password.json
{ "credential":"password"}
```

Note

The Oracle GoldenGate administrator account user name registered with XAG, must be the same for the Service Manager and all deployments managed by the Service Manager. This also applies to the administrator account password, it must be the same for the Service Manager and all deployments.

3. Change the password with XAG, as the root user:

```
# agctl modify goldengate instance_name
--adminuser OGG admin user
Enter password for 'oggadmin' :
```

4. Re-enable the XAG healthcheck monitoring:

```
$ agctl enable goldengate instance_name
```

Manage Certificates for Deployments

Learn about managing certificates for deployments.

Add Client Certificate

To add a client certificate:

1. Click the plus (+) sign next to the **Client Certificates** section. The Add Client Certificate dialog box appears.
2. Enter the following details for the client certificate:
 - **Unique Name:** Name of the certificate.
 - **Certificate PEM:** Enter a certificate .pem file or upload a .pem file.
 - **Private-Key PEM:** Enter or upload the private key for the .pem file.
 - **CA Certificates:** Enter or upload the CA certificate.
3. Click **Add**.

Add a CA Certificate

To add a CA certificate:

1. Click the plus (+) sign next to **CA Certificates**. The Add CA Certificate dialog box appears.
2. Enter the following details for the CA certificate:
 - **Unique Name** for the CA certificate.
 - **Certificate PEM** value can be entered in the box or uploaded.
 - **Certificate location** can be shared. CA Certificates for the Service Manager are always shared and cannot be local. When adding or replacing CA certificates, the Shared option is always force-checked.
3. Click **Add**.

Apply Certificates to an Oracle GoldenGate Deployment

Certificates can apply to:

- **A specific deployment:** These certificates are local to the deployment. See [Create RootCA and Server Certificates](#).
- **Shared across deployments added to the same Service Manager:** These are shared certificates created from the Service Manager Certificate Management page. These certificates can be shared across multiple deployments supervised by one Service Manager.
- **Different source and target deployments:** These are called external certificates (extern) with different source and target deployments. See [Create External Trusted RootCA and Distribution Client Certificates](#).

Note

Adding a non-CA self-signed certificate as a trusted certificate using Certificate Management page's CA Cert section is not supported and will result in an error.

Check Details of Certificates Used in a Deployment

To check the details of a certificate including start date and expiry date for any certificate:

1. Log in to the Service Manager.
2. Select **Certificate Management** from the left navigation pane of the Service Manager home page.
3. Select the deployment from the drop-down list to view information about the server, client, and CA certificates. The validity period (expiration date) of the certificate along with the used signing algorithms from the issuer are displayed. You can also view the certificates available with the Service Manager by selecting Service Manager from the Certificate Management sub-menu in the left navigation pane.

The screenshot shows the Oracle GoldenGate Service Manager interface. The left navigation pane includes sections for 'scott Security' (Home, User Administration, Deployments, Certificate Management) and 'phoenix433404' (ServiceManager, Diagnosis Log, Debug Log). The main content area displays three sections: 'Server Certificates', 'Client Certificates', and 'CA Certificates'. Each section contains a table with columns for Status, Name, Issued To, Issued By, Validity Start, Validity Expiration, and Action.

Server Certificates +						
Status	Name	Issued To	Issued By	Validity Start	Validity Expiration	Action
✓	default	CN = phoenix433404	CN = RootCA	2024-02-12T13:26:46Z	2025-02-21T13:26:46Z	[Edit] [Details]

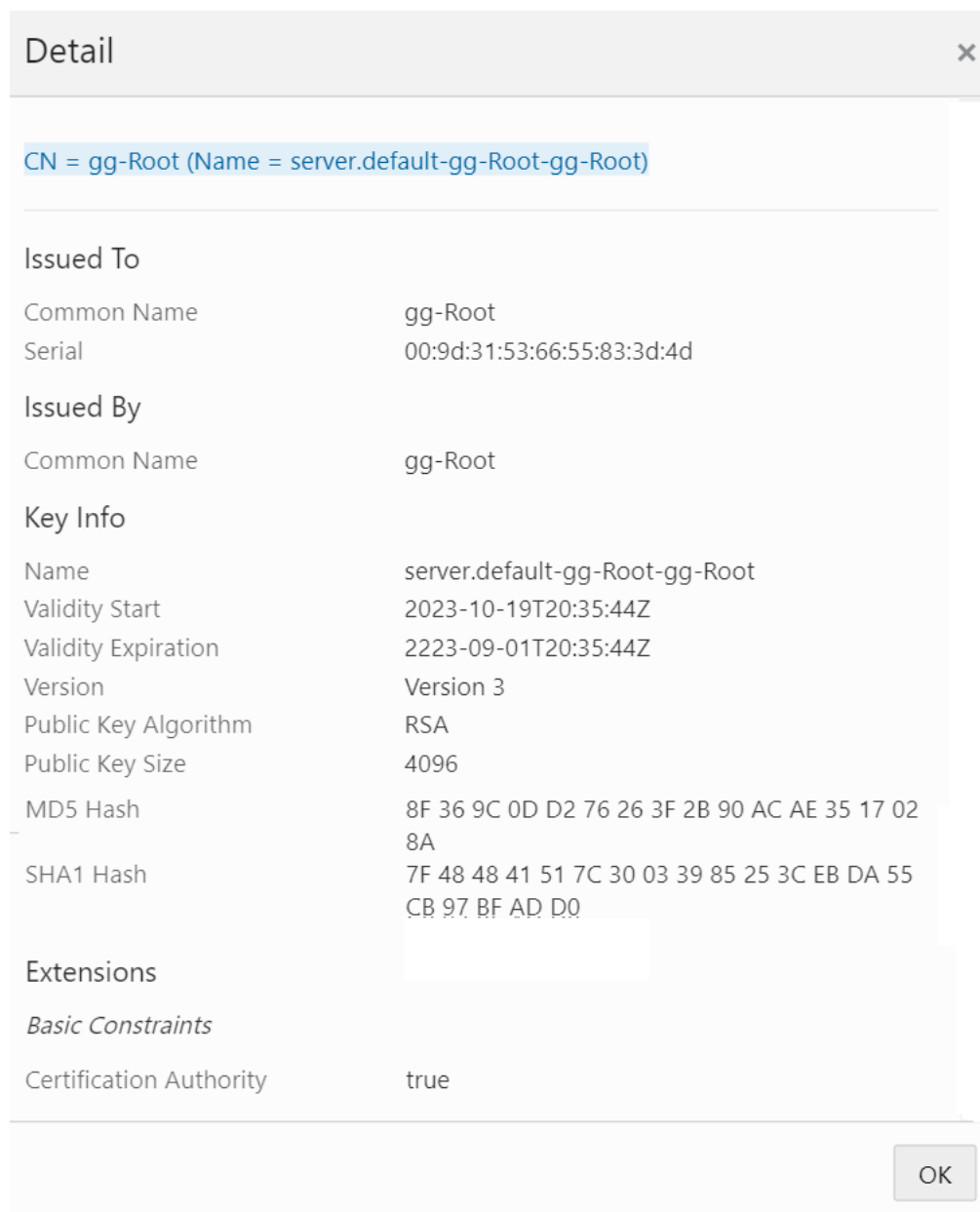
Client Certificates +						
Status	Name	Issued To	Issued By	Validity Start	Validity Expiration	Action
✓	default	CN = phoenix433404	CN = RootCA	2024-02-12T13:26:53Z	2025-02-21T13:26:53Z	[Edit] [Details] [Delete]

CA Certificates +						
Shared						
Status	Name	Issued To	Issued By	Validity Start	Validity Expiration	Action
✓	8a72162e4047	CN = RootCA	Self-signed	2024-02-12T13:26:42Z	2044-02-07T13:26:42Z	[Details] [Edit] [Delete]

Local

To retrieve certificate information using the REST API, see [Certificates](#) for details in the *Oracle GoldenGate REST API Documentation*.

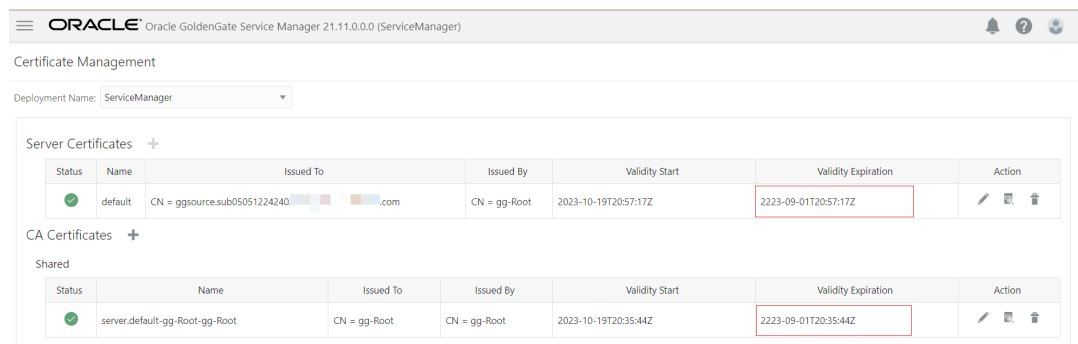
4. To view further details about a certificate, click the **Details** icon from the **Action** column. The certificate details are displayed, shown in the following image:



Replace Certificates in a Deployment

You cannot renew a certificate. You can only replace it with a new certificate. Make sure to check the expiry details of certificates that you intend to replace. Use the following steps to replace certificates:

1. Click the **Certificate Management** tab from the left navigation pane of the Service Manager.
2. Select the deployment from the drop-down list to view information about the server, client certificates, and CA certificates.
3. Click the **Detail** icon from the **Action** column of the certificate store table to view details about the certificate including certificate start and expiration dates, shown in the following image:



4. Click the **Replace** (pencil) icon to replace server certificates.
5. Click the **Delete** icon in the **Action** column to delete the certificate.

Delegate User Management to an External Identity Provider

Learn about delegating user authentication and authorization to external identity providers.

Oracle GoldenGate supports IDCS and IAM as cloud-based identity providers and OAM as an on-premise identity provider. Starting with Oracle GoldenGate 23.9, Microsoft Azure EntraID is also available as an external identity provider.

By using identity providers, you delegate the Oracle GoldenGate user management and authentication to the OpenID-complaint external identity providers. To utilize this feature, you can create authorization profiles in Oracle GoldenGate while registering the application in the corresponding external identity providers.

An authorization profile created in Oracle GoldenGate allows integration with external identity providers (IdPs), which can be configured in Oracle GoldenGate using Authorization Profiles. External IdPs provide user management (using users, groups, and alignment between users, groups, and applications) capabilities. To set up a connection between an external IdP and Oracle GoldenGate, a confidential application needs to be created using OAuth2. From this confidential application, Oracle GoldenGate derives the Client ID and Client Secret for authenticating the IdP system.

The external IdP system gets the information including the redirect URIs and post-logout URLs from Oracle GoldenGate. This allows managing Oracle GoldenGate user access through external servers instead of creating users for accessing Oracle GoldenGate.

Note

A prerequisite for setting up authorization profiles is to have a secure deployment. The deployment can be secured using Server certificates or a Reverse Proxy configuration.

Configure IAM for Oracle GoldenGate

Identity and Access Management (IAM) uses identity domains to provide identity and access management features such as authentication, single sign-on (SSO), and identity lifecycle management for Oracle Cloud as well as for Oracle and non-Oracle applications, including SaaS, cloud hosted, or on premises.

To set up an IAM application, perform the following tasks on the IAM side:

1. Create a domain to set up the application.
2. Create a user account for a user in an OCI IAM identity domain. See [Create Users](#) in OCI Documentation.
3. Create a group. A group has no permissions until you do one of the following:
 - Write at least one policy that gives that group permission to either the tenancy or a compartment. When writing the policy, you can specify the group by using either the unique name or the group's OCID. For information about writing policies, see [Managing Policies](#).
 - Assign the group to an application.See [Create Groups](#),
4. Add an application. Confidential applications run on a protected server. See [Add Applications](#)

Integrating your applications makes it easy for users to sign in with single sign on and gives you a central place to manage their permissions. Application integration includes securing your users, protecting the resources within the applications, and enabling users to access your applications through single sign-on (SSO). Confidential applications run on a protected server.
5. Configure [OAuth](#) to protect resources of the confidential application. Authorized resources define the way a client can access the resources in a Confidential application. Specify the following values to set up OAuth for Oracle GoldenGate. The values that are specified here will be used when creating the authorization profile in Oracle GoldenGate.
 - Access token expiration (seconds): This defines how long the access token associated with your confidential application remains valid.
 - The primary audience (recipient): This is where the access token for the confidential application is processed.
 - Scopes: These are used to specify which parts of other applications that you want your application to access. The scope is fixed (**oggServiceToService**)
6. Add Application Roles.
7. Add Resources.
8. Activate Application.
9. After configuring the confidential application in IAM, create the authorization profile in Oracle GoldenGate. See [Create an Authorization Profile](#).

For an example of how the Authorization Profile would be configured with the values from the IAM application, see [Example IAM Application and Oracle GoldenGate Authorization Profile Configured for an IAM Application](#).

Example IAM Application and Oracle GoldenGate Authorization Profile Configured for an IAM Application

The following examples show samples of the IAM confidential application and how its values are used when the Oracle GoldenGate Authorization Profile is created.

Example: IAM Application Configuration

The following example shows the IAM configuration for the confidential application. Notice the HTTPS address from the *jwt-uri* that shows additional key information:

```

IAM                : TestDemo_IAM
Domain             : TestDemo_Domain
Groups             : GG_Group_Security, GG_Group_Administrator,
GG_Group_Operator, and GG_Group_User
Application        : GG_TestDemo_App
Audience          : GG_TestDemo_PrimaryAudience
Scope             : urn:ogg:serviceToService

RedirectURI        : https://east.oraclevcn.com:8231/services/v2/
authorization
Post-logout URL    : https://east.oraclevcn.com:8231/ojrsignout

```

Example: Oracle GoldenGate Authorization Profile

Using the values shown in Example 1, create an authorization profile in Oracle GoldenGate, which would have the following configuration:

```

Profile Type       : IAM
Profile Name       : GG_TestDemo_IAM
Description        : Authorization Profile for TestDemo using IAM
Tenant Discovery URI: https://idcs-[...].identity.oraclecloud.com/.well-known/
openid-configuration
ClientID           :
123abc456def789ghi012jkl345mno67
Client Secret      : xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx
Group Mapping      : GG_Group_Security, GG_Group_Administrator,
GG_Group_Operator, and GG_Group_User

```

Configure OAM for Oracle GoldenGate

To configure Oracle Access Manager (OAM) with Oracle GoldenGate, perform the following tasks on the OAM side:

1. Configure OAM within Weblogic Server.
2. Create Users. The OAM Admin user needs to create these users.
3. Create Group. The OAM Admin user needs to create the group.
4. Create an Identity Domain.
5. Create a Resource Server. For the Resource Server, specify the following values:
 - Domain Name: This can be any name.
 - Resource Server Name: This can be any name.
 - Scope: This name is fixed. The value is **oggServiceToService**.
6. Create an Application Server with following values:
 - Server and ID
 - Scope (Resource Server Name)

- Redirect URIs: Although redirect URIs are optional, they must be the same as the Oracle GoldenGate URIs

This is the link to the Oracle GoldenGate Service. For example:
`https://OGGServerName:ServerPort:/services/v2/authorization`

7. Configure the authorization profile for OAM on the Oracle GoldenGate side. See [Create an Authorization Profile](#).

For samples of setting up the OAM application and creating an Authorization Profile in Oracle GoldenGate using the OAM application values, see [Example of OAM Application Configuration and Oracle GoldenGate Authorization Profile](#).

Example of OAM Application Configuration and Oracle GoldenGate Authorization Profile

The following scripts are examples of configuring various components when creating a secure Oracle Access Manager (OAM) application.

Creating an Identity Domain

The following example shows the creation of an Identity Domain in OAM.

```
curl -x ""
-u 'weblogic:weblogic1'
'http://east.oraclevcn.com:18585/oam/services/rest/ssa/api/v1/
oauthpolicyadmin/oauthidentitydomain'
-H 'Content-Type: application/json'
-d '
  {"name": "OGGDomain1"
  , "identityProvider": "OUD"
  , "description": "Test Domain"
  , "tokenSettings":
    [{"tokenType": "ACCESS_TOKEN"
    , "tokenExpiry": 3600
    , "lifeCycleEnabled": false
    , "refreshTokenEnabled": false
    , "refreshTokenExpiry": 86400
    , "refreshTokenLifeCycleEnabled": false
    }
  , {"tokenType": "AUTHZ_CODE"
  , "tokenExpiry": 3600
  , "lifeCycleEnabled": false
  , "refreshTokenEnabled": false
  , "refreshTokenExpiry": 86400
  , "refreshTokenLifeCycleEnabled": false
  }
  , {"tokenType": "SSO_LINK_TOKEN"
  , "tokenExpiry": 3600
  , "lifeCycleEnabled": false
  , "refreshTokenEnabled": false
  , "refreshTokenExpiry": 86400
  , "refreshTokenLifeCycleEnabled": false
  }
  ]
  , "errorPageURL": " http://east.oraclevcn.com:2222/oam/pages/
servererror.jsp"
```

```

    , "consentPageURL": "http://east.oraclevcn.com:2222/oam/pages/
consent.jsp"
    , "customAttrs": null
  }'

```

Create Resource Server

The following example shows creating a Resource Server for the OAM application:

```

curl -x ""
    -u weblogic:weblogic1
    -H "Content-Type: application/json" '
    http://east.oraclevcn.com:18585/oam/services/rest/ssa/api/v1/
    oauthpolicyadmin/application'
    -X POST
    -d '
    {
      "idDomain": "OGGDomain1"
      , "name": "OGGResourceServerCorrect"
      , "scopes":
      [
        {
          "scopeName": "oggServiceToService"
          , "description": "test ogg urn scope"
        }
      ]
    }'

```

Creating an Application

The following example shows creating an Application with some redirect URIs, using OAM:

```

curl -x ""
    -v
    -u weblogic:<weblogicpwd>
    -H "Content-Type: application/json"
    'http://east.oraclevcn.com:18585/oam/services/rest/ssa/api/v1/
    oauthpolicyadmin/client'
    -X POST
    -d '
    {
      "secret": "OGGClient1Secret"
      , "id": "OGGClientID1"
      , "scopes": ["OGGResourceServerCorrect.oggServiceToService"]
      , "clientType": "CONFIDENTIAL_CLIENT"
      , "idDomain": "OGGDomain1"
      , "description": "Client Description"
      , "name": "OGGClientCorrect"
      , "grantTypes": ["PASSWORD", "CLIENT_CREDENTIALS", "AUTHORIZATION_CODE"]
      , "defaultScope": "OGGResourceServerCorrect.oggServiceToService"
      , "redirectURIs":
      [
        {
          "url": "https://localhost:9010/services/v2/authorization",
          "isHttps": true
        }
        , {
          "url": "https://west.oraclevcn.com:9010/services/v2/
          authorization", "isHttps": true
        }
        , {
          "url": "https://west.oraclevcn.com:9001/services/v2/
          authorization", "isHttps": true
        }
        , {
          "url": "https://localhost:9001/services/v2/authorization",
          "isHttps": true
        }
      ]
    }'

```

```
, "attributes": [{"attrName": "OGGGroups"
, "attrValue": "${user.groups}"
, "attrType": "STATIC"
}]'
```

Create an Authorization Profile in Oracle GoldenGate for the OAM Application

The following code snippet is a sample of how to create an authorization profile in Oracle GoldenGate (using cURL) for the OAM application:

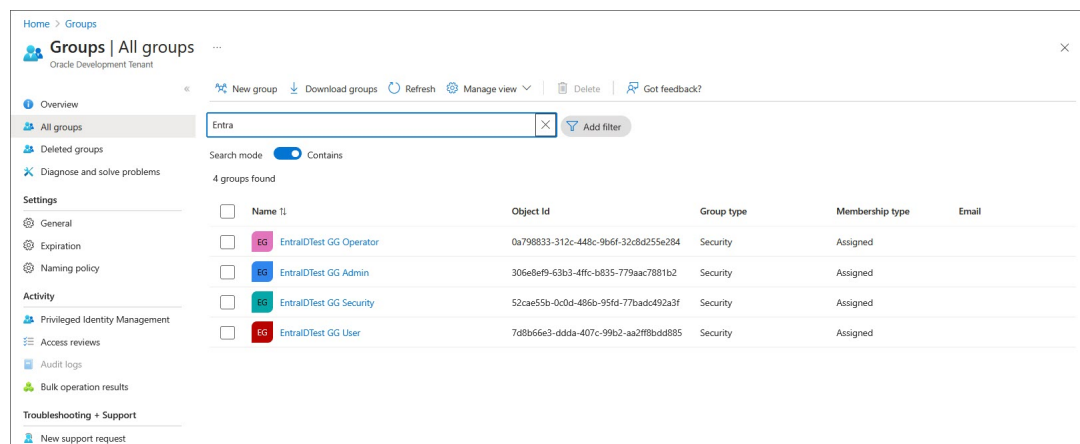
```
curl -k
-u ggma:<ggmapwd>
-X POST
-d '{"type": "oam"
, "clientId": "OGGClientID1"
, "clientSecret": "OGGClient1Secret"
, "tenantDiscoveryURI": "http://east.oraclevcn.com:2222/.well-known/
openid-configuration"
, "groupToRoles": {"securityGroup": "fusionapps_gg_prod_sec"
, "administratorGroup": "fusionapps_gg_prod_admin"
}
, "identityDomain": "OGGDomain1"
}'
https://localhost:9010/services/v2/deployments/ServiceManager/
authorization/profiles/smProfile
```

Configure Microsoft Azure EntraID for Oracle GoldenGate

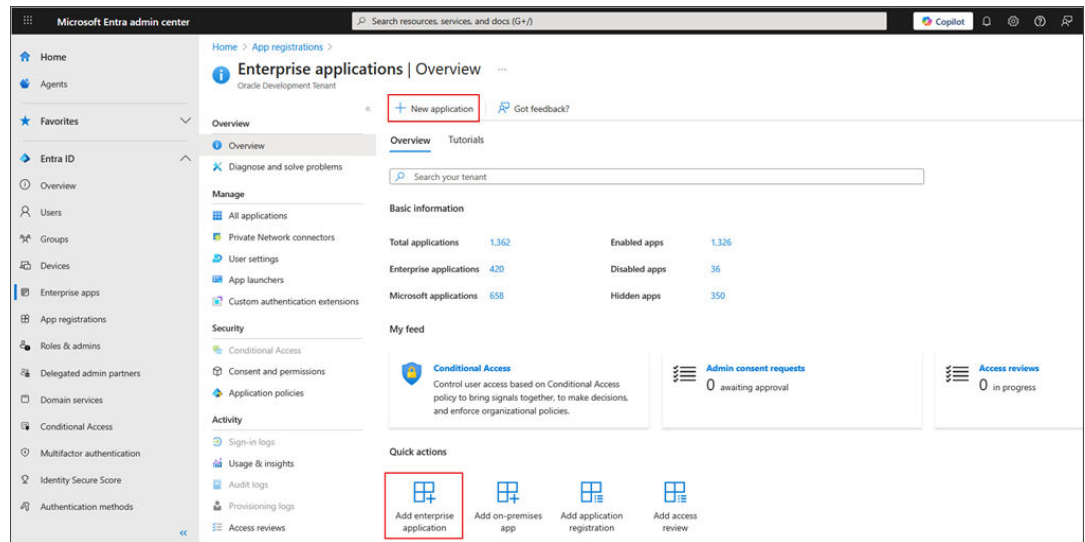
When creating the Authorization Profile, Oracle GoldenGate maps the Entra ID security groups into four roles for authorization: Security, Administrator, Operator, User. The Entra ID user who belongs to the group has the corresponding mapped role in Oracle GoldenGate.

On the Microsoft Azure EntraID portal, perform the following tasks to configure the user groups that would map with Oracle GoldenGate:

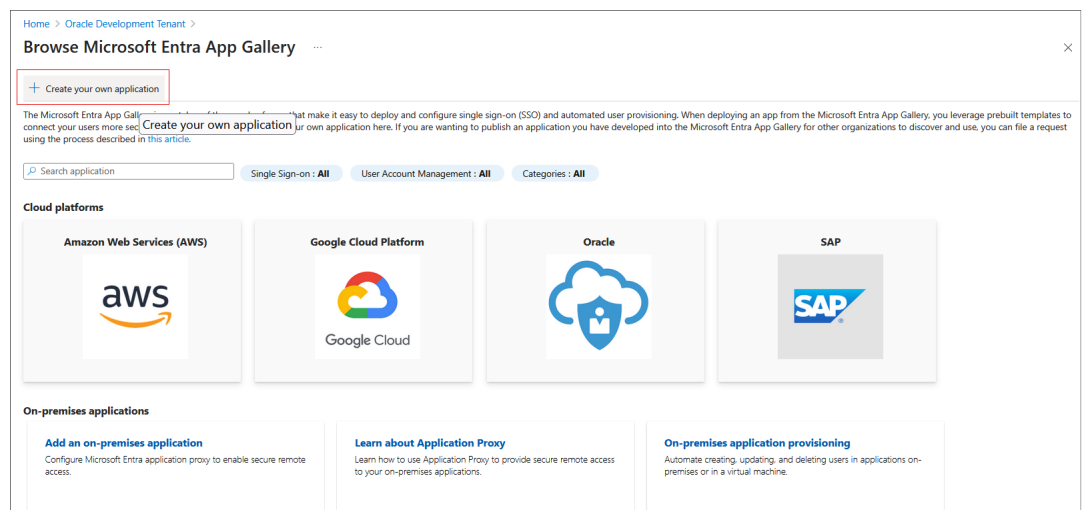
1. Create the appropriate user and security groups in EntraID. You will need four security Groups that will be mapped to the respective Oracle GoldenGate user roles (Security, Admin, Operator, User). The group name could be any name you choose.



2. Select **Add, Enterprise Application** from the Overview page.



3. Click **Create your own application**.



4. Name your application, select **Register an application to integrate with Microsoft Entra ID** and click **Create**.

Create your own application ✕

 Got feedback?

If you are developing your own application, using Application Proxy, or want to integrate an application that is not in the gallery, you can create your own application here.

What's the name of your app?

OGG Auth Application 

What are you looking to do with your application?

☐ Configure Application Proxy for secure remote access to an on-premises application

☒ Register an application to integrate with Microsoft Entra ID (App you're developing)

☐ Integrate any other application you don't find in the gallery (Non-gallery)

Create

5. To register the application, select **Accounts in this organizational directory only (... - Single tenant)**.

Home > Oracle Development Tenant > Browse Microsoft Entra App Gallery >

Register an application

* Name

The user-facing display name for this application (this can be changed later).

OGG Auth Application

Supported account types

Who can use this application or access this API?

☒ Accounts in this organizational directory only (Oracle Development Tenant only - Single tenant)

☐ Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant)

☐ Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)

☐ Personal Microsoft accounts only

[Help me choose...](#)

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Select a platform

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from [Enterprise applications](#).

By proceeding, you agree to the [Microsoft Platform Policies](#)

[Register](#)

- Go back to Entra ID **Overview** page, under **Manage**, select **Enterprise applications** and search for the application name you created in the previously.
- Under **Manage**, select **Users and groups**. Click **Add user/group**.

Home > Oracle Development Tenant > Browse Microsoft Entra App Gallery > Oracle Development Tenant > Manage tenants > Oracle Development Tenant > Enterprise applications | All applications > OGG Auth Application

OGG Auth Application | Users and groups

Enterprise Application

[+ Add user/group](#) [Edit assignment](#) [Remove assignment](#) [Update credential](#) [Refresh](#) [Manage view](#) [Got feedback?](#)

Overview

Deployment Plan

Diagnose and solve problems

Manage

Properties

Owners

Roles and administrators

Users and groups

Single sign-on

Provisioning

Application proxy

Self-service

Custom security attributes

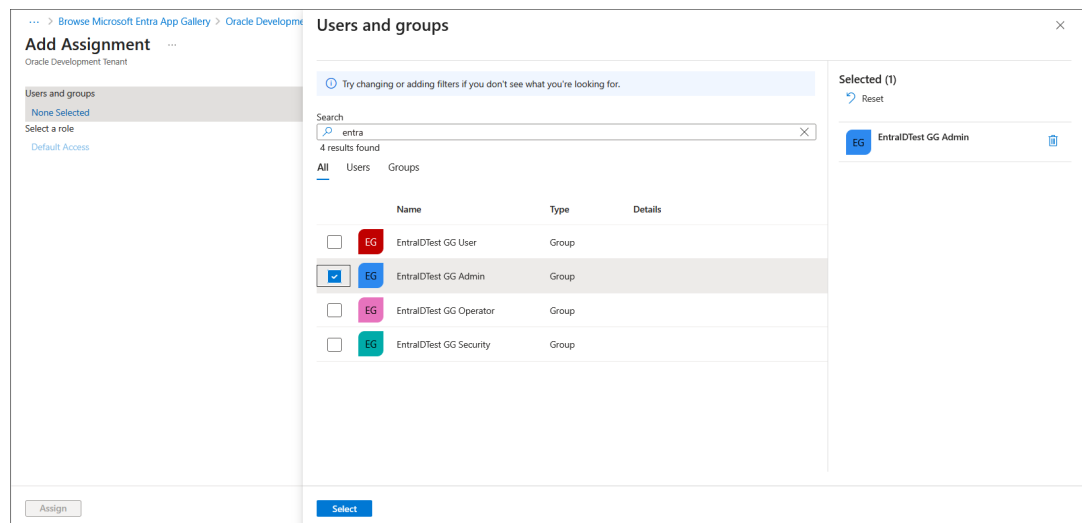
The application will not appear for assigned users within My Apps. Set 'visible to users?' to yes in properties to enable this.

Assign users and groups to app-roles for your application here. To create new app-roles for this application, use the [application registration](#)

First 200 shown, search all users & groups

Display name	Object type	Role assigned
No application assignments found		

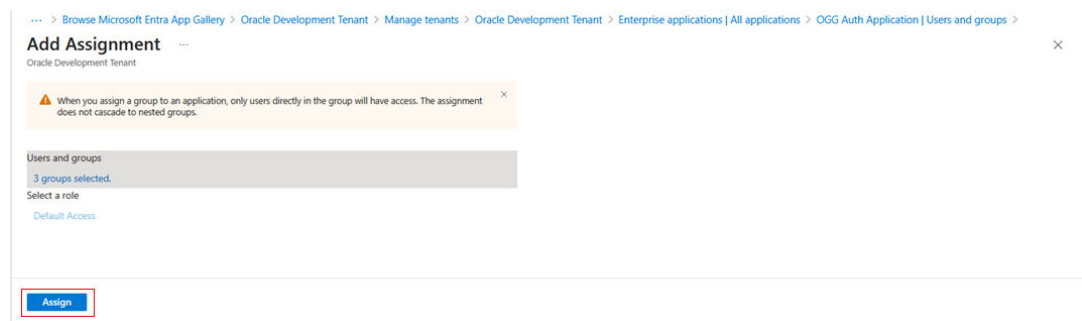
- Search for and select the groups to assign to this application and ensure that the selected groups appear in the right-hand pane.



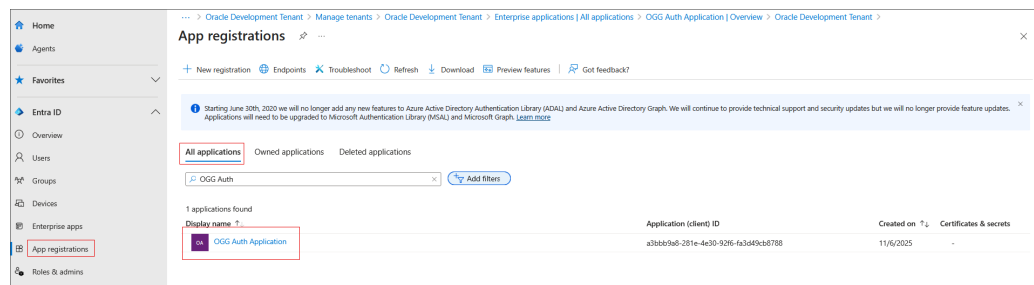
9. Select all groups, then click **Select**, and click **Assign**.

Note

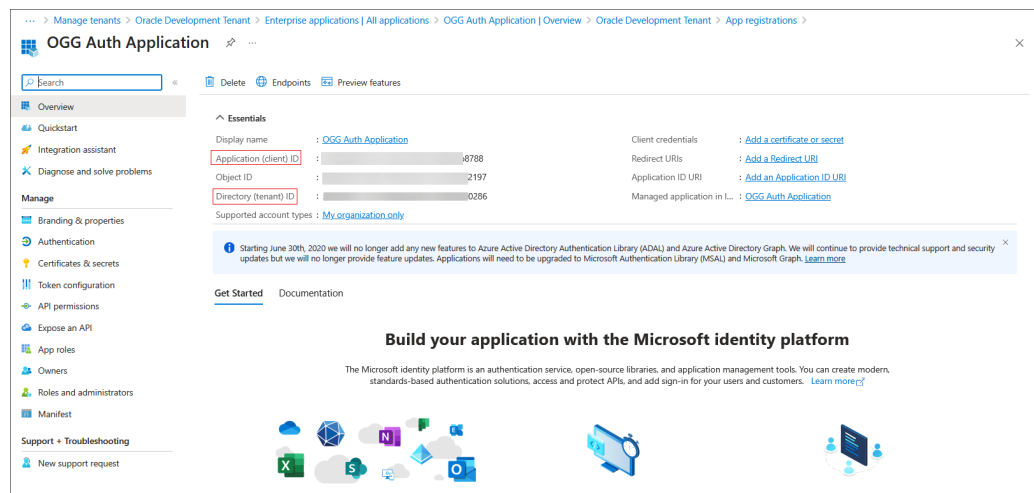
Make sure you assign all the Entra ID security groups created in previous steps to this application.



10. (Optional) Assign application owners by following these steps:
- Under Manage select **Owners**.
 - Click **Add**.
 - Search for and select owners.
 - Make sure that selected owners appear on the right-hand pane.
11. Register Application by creating client secrets:
- Go back to Microsoft Entra ID Overview page
 - Select **App registrations** and then select **All Applications**.



- c. Search for and select the application name created in step 2.
- d. Now, the application shows the details. Note down the **Application (client) ID** and **Directory (tenant) ID**.

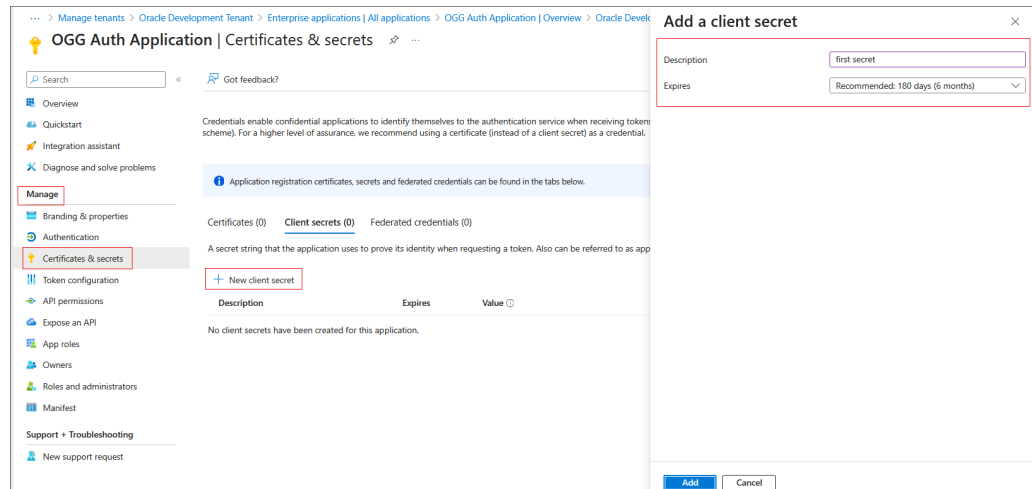


Note the Client ID, Tenant ID, and Client Secret values. These values will be used while configuring the authorization profile in Oracle GoldenGate.

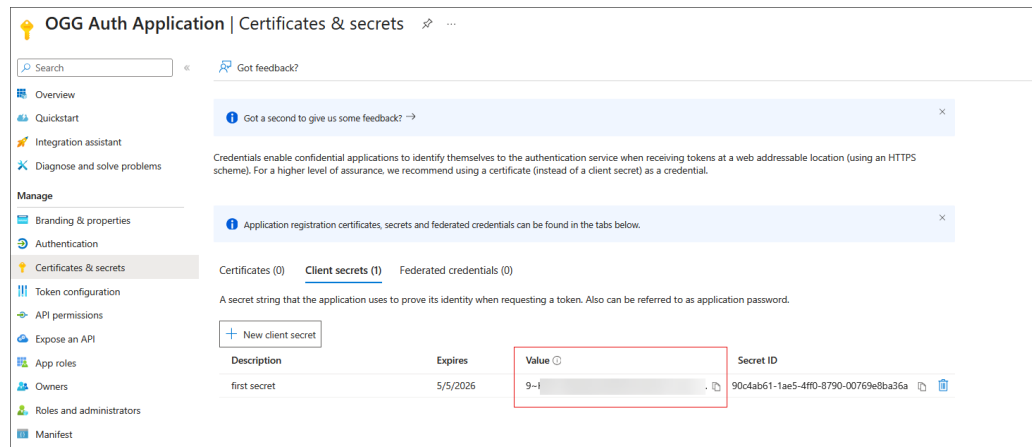
The Tenant ID is used to construct Tenant Discovery URI, as shown in the following example:

`https://login.microsoftonline.com/{tenantID}/v2.0/.well-known/openid-configuration`

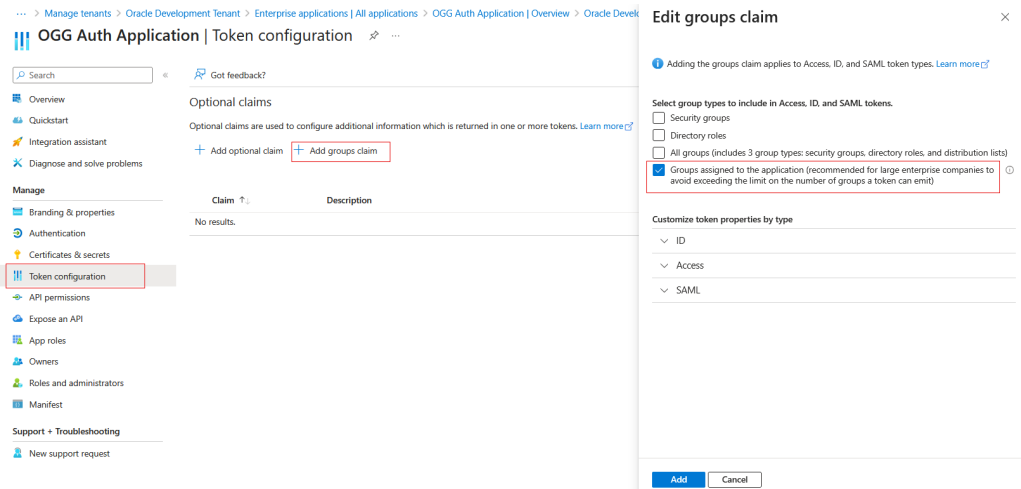
- e. Under **Manage**, select **Certificates & secrets**
- f. Select **Client secrets** and then click **New client secret**.



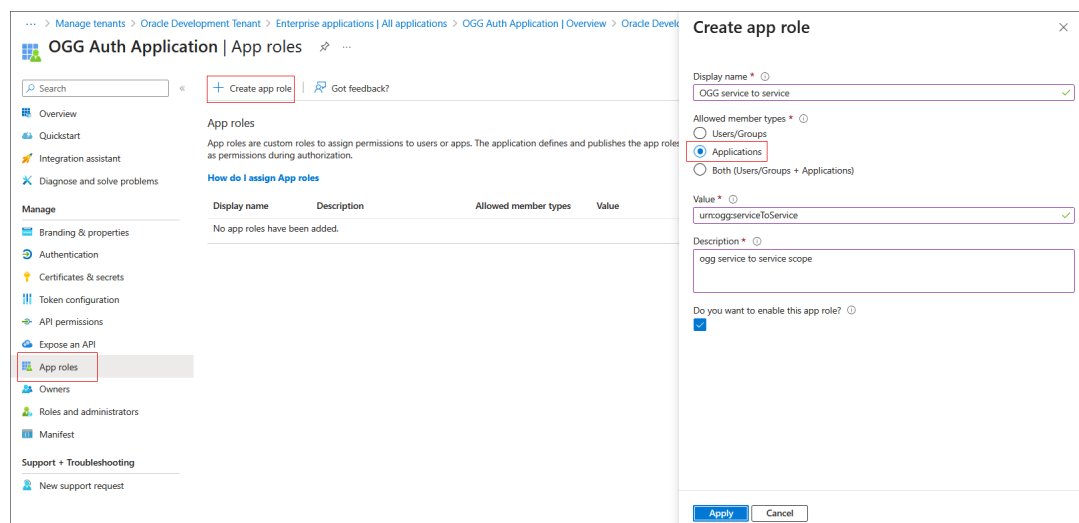
- g. Add a description and select an expiration period.
- h. Click **Add**. Note down the client secret **Value**.



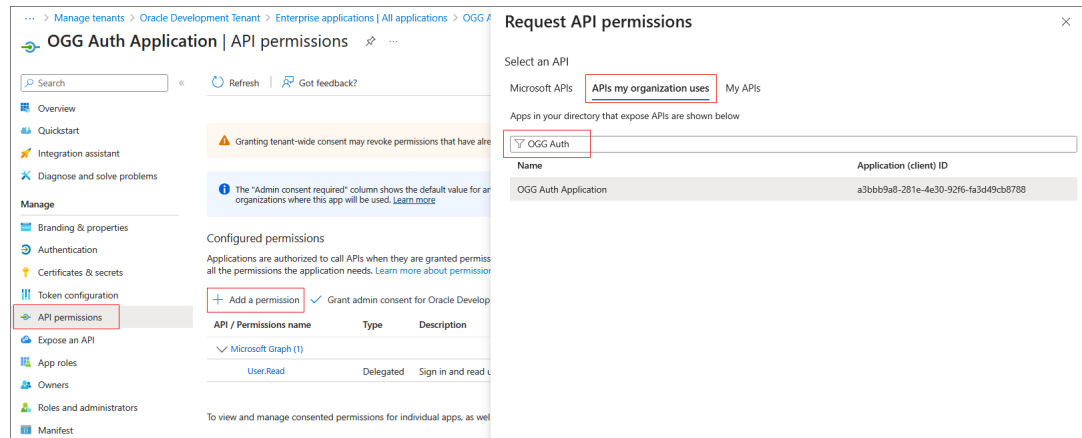
12. Create the token configuration for the OGG Application:
 - a. Under **Manage** in the navigation pane, choose Token configuration.
 - b. Click **Add groups claim** and select **Groups assigned to the application (...)** and click **Add**.



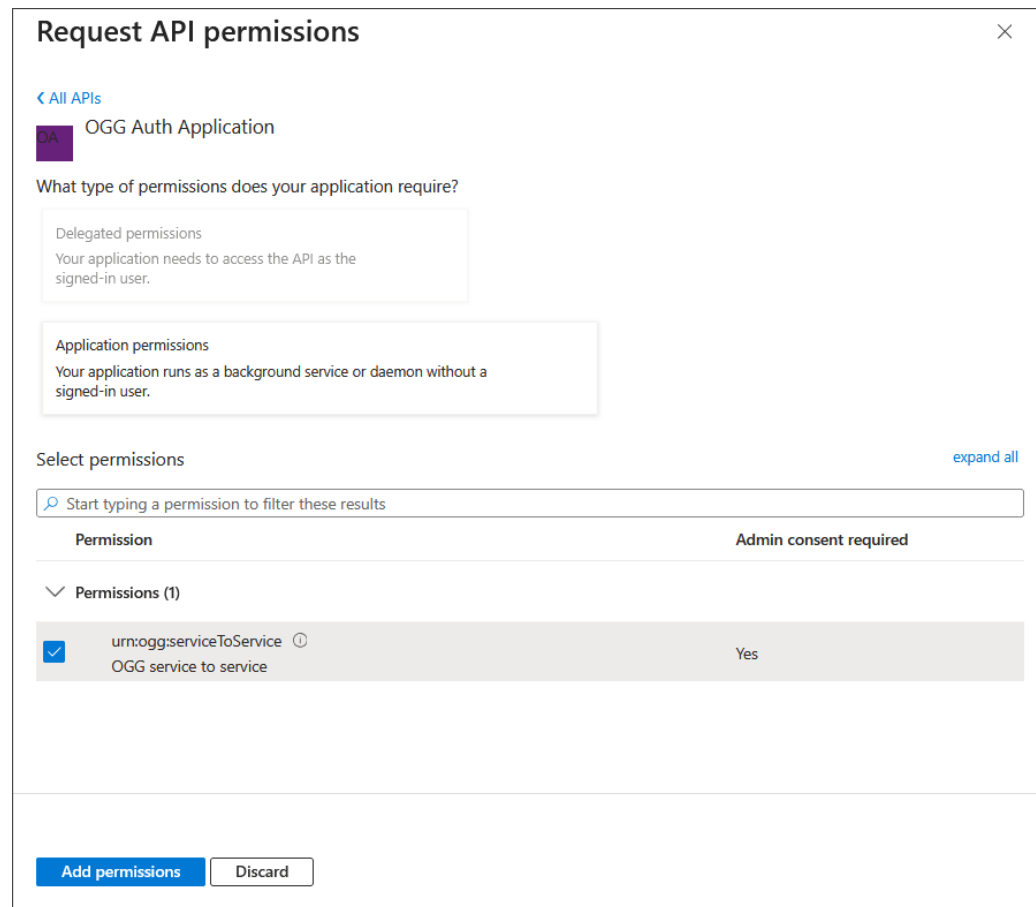
13. Create GoldenGate Service App roles:



- a. Under **Manage** in the navigation pane, choose **App Roles**.
 - b. Click **Create app role** and specify a display name.
 - c. Under **Allowed member types** select **Applications**.
 - d. Set **Value** to **urn:ogg:serviceToService** and fill the description.
 - e. Select **Do you want to enable this app role?** and click **Apply**.
14. Add API Permission to the GoldenGate Service App Roles and grant admin consent:



- a. Under **Manage**, select **API permissions**.
- b. Click **Add a permission**.
- c. Click **APIs my organization uses**.
- d. Search for the application created in Step 2.
- e. Click Application permissions.
- f. Select **urn:ogg:serviceToService** and click **Add permissions**.



- g. The permission should appear under **Configured permissions**.

Configured permissions

Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. [Learn more about permissions and consent](#)

+ Add a permission ✓ Grant admin consent for Oracle Development Tenant

API / Permissions name	Type	Description	Admin consent requ...	Status
Microsoft Graph (1)				...
User.Read	Delegated	Sign in and read user profile	No	...
OGG Auth Application (1)				...
urn:ogg:serviceToService	Application	OGG service to service	Yes	⚠ Not granted for Oracle ...

To view and manage consented permissions for individual apps, as well as your tenant's consent settings, try [Enterprise applications](#).

- h. Click on **urn:ogg:serviceToService**.
- i. In the pop-up, notice that Admin consent required is set to **Yes**.

urn:ogg:serviceToService ✕

OGG Auth Application

🗑 Remove permission

Resource App ID

a3bbb9a8-281e-4e30-92f6-fa3d49cb8788 📋

Permission ID

ff6c360b-09f4-4d02-bf4f-ba99f1befcaf 📋

Admin consent required

Yes

Display Name

OGG service to service

Description

ogg service to service scope

15. Add **Microsoft Graph Application.ReadWrite.All** API permissions to the Application and grant Admin consent.
- a. Click **Add a permission** and then select **Microsoft Graph** from the pop-up window.

The screenshot shows the OICS console with a 'Successfully granted admin consent' message. Below, the 'Configured permissions' section shows a table with columns for API/Permissions name, Type, and Description. A red box highlights the '+ Add a permission' button. To the right, the 'Select an API' dialog is open, showing 'Microsoft Graph' as the selected API, with a red box around its description.

b. Select **Application permissions**.

The 'Request API permissions' dialog shows the 'Microsoft Graph' API selected. Under 'What type of permissions does your application require?', the 'Application permissions' option is selected and highlighted with a blue box. Below, the 'Select permissions' section shows a search bar with 'Application.ReadWrite.All' entered. A table lists the selected permission: 'Application.ReadWrite.All' with the description 'Read and write all applications' and 'Admin consent required' set to 'Yes'. At the bottom, there are 'Add permissions' and 'Discard' buttons.

- c. Search for and select **Application.ReadWrite.All**.
- d. Click **Add permissions**.
- e. Perform the same steps done to grant consent for **urn:ogg:serviceToService** but for **Application.ReadWrite.All**.

16. Manually update the token request properties in the Manifest:

The following options are not available under other menus, so they must be set manually, as shown in the following image:

```

"optionalClaims": {
  "accessToken": [
    {
      "additionalProperties": ["cloud_displayname"],
      "essential": false,
      "name": "groups",
      "source": null
    }
  ],
  "idToken": [
    {
      "additionalProperties": ["cloud_displayname"],
      "essential": false,
      "name": "groups",
      "source": null
    }
  ],
  "saml2Token": [
    {
      "additionalProperties": ["cloud_displayname"],
      "essential": false,
      "name": "groups",
      "source": null
    }
  ]
},

```

Manually updating the token request in Manifest.

Under **Manage**, click **Manifest**.

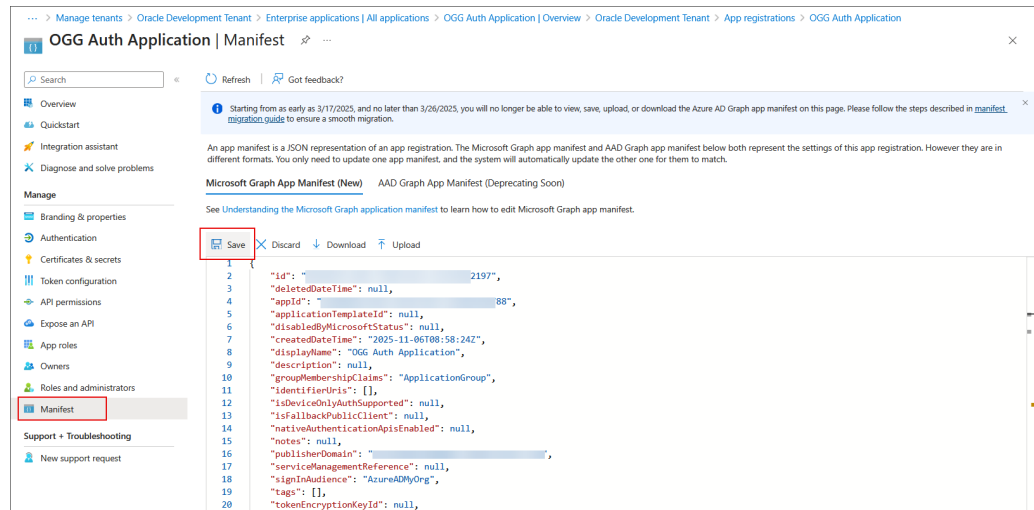
- a. Under the JSON `/optionalClaims/accessToken/additionalProperties` add the value `cloud_displayname`.
- b. Under the JSON `/optionalClaims/idToken/additionalProperties`, add the value `cloud_displayname`.
- c. Under the JSON `/optionalClaims/saml2Token/additionalProperties` add the value `cloud_displayname`.
- d. Under the JSON `/api/requestedAccessTokenVersion` set value to 2.

```

"api": {
  "acceptMappedClaims": null,
  "knownClientApplications": [],
  "requestedAccessTokenVersion": 2,
  "oauth2PermissionScopes": [],
  "preAuthorizedApplications": []
},

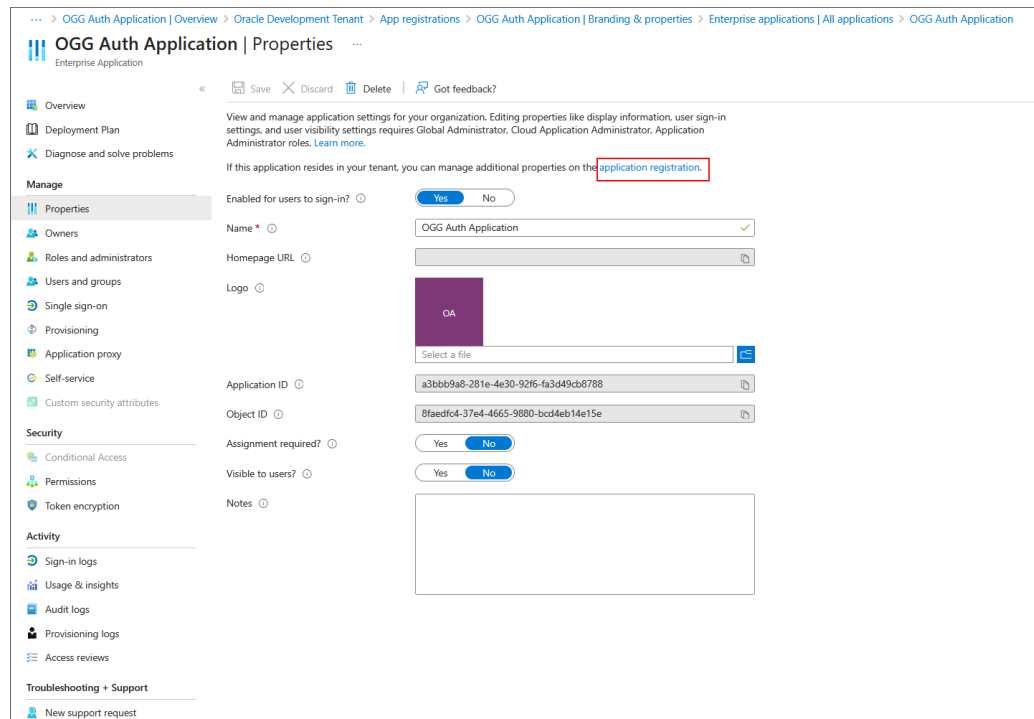
```

- e. Save the Manifest.

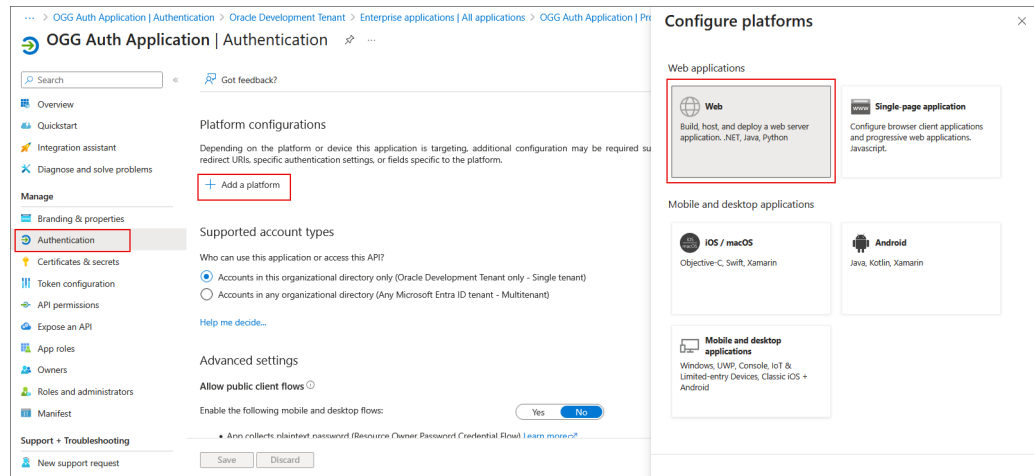


17. (Optional) Create the redirect URL for Application Authentication.

- Under **Microsoft Entra ID, Enterprise applications**, choose the application you created in Step 2.
- Under **Manage**, click **Properties** and then click the **application registration** link.



- Under **Manage**, click **Authentication** and then click **Add a platform**.
-
- Click **Web**.



18. In the Configure Web dialog box, enter **Redirect URIs.**

For the Service Manager deployment, you should add the path `/services/v2/` authorization for the Service Manager.

For each Microservices deployment, you should add the path `/services/v2/` authorization for the Administration Service.

Configure Web

[All platforms](#)
[Quickstart](#)
[Docs](#)

*** Redirect URIs**

The URIs we will accept as destinations when returning authentication responses (tokens) after successfully authenticating or signing out users. The redirect URI you send in the request to the login server should match one listed here. Also referred to as reply URIs. [Learn more about Redirect URIs and their restrictions](#)

✓

Front-channel logout URL

This is where we send a request to have the application clear the user's session data. This is required for single sign-out to work correctly.

✓

Implicit grant and hybrid flows

Request a token directly from the authorization endpoint. If the application has a single-page architecture (SPA) and doesn't use the authorization code flow, or if it invokes a web API via JavaScript, select both access tokens and ID tokens. For ASP.NET Core web apps and other web apps that use hybrid authentication, select only ID tokens. [Learn more about tokens](#).

Select the tokens you would like to be issued by the authorization endpoint:

☐ Access tokens (used for implicit flows)
 ☐ ID tokens (used for implicit and hybrid flows)

Configure

Cancel

19. Click **Configure**.

Next step is to download the RootCA certificate for Microsoft Azure EntraID and add it under the Service Manager Shared RootCA certificates.

After you set up the Root Certificate Authority, you can configure the Oracle GoldenGate Authorization Profile using the values created in this topic. See [Create an Authorization Profile](#).

Add Root CA Certificate for Microsoft EntraID Azure

Before you start creating an authorization profile in Oracle GoldenGate, you need to add a Root Certificate Authority for Azure. Add Root CA Certificate as Trusted Certificates to the Oracle GoldenGate deployment:

- Download the Azure Root CA Certificates from the following location:
<https://learn.microsoft.com/en-us/azure/security/fundamentals/azure-ca-details?tabs=root-and-subordinate-cas-list#root-certificate-authorities>
 You need only the Root Certificate Authorities.
- Convert the certificates downloaded from crt to pem format by using the openssl command in any Linux operating system or Mac OS.

```
openssl x509 -in <cert_file> -out <pem_file_name> -outform pem
```

For example:

```
openssl x509 -in DigicertRootGlobalCA.crt -out DigicertRootGlobalCA.pem -  
outform pem
```

Assuming you are using Oracle GoldenGate on-premises or OCI Marketplace GoldenGate images, you must upload those Root CA certificates as trusted certificates through the Service Manager web interface.

3. Find the Service Manager under **Certificate Management** menu, click **CA Certificates +**.
4. Click the file upload button to upload the converted .pem file.

Add CA Certificate X

Add the CA Certificate

Unique Name
MS ECC Root

Certificate PEM

-----BEGIN CERTIFICATE-----
MIIDrzCCApegAwIBAgIQCDvgVpBCRrGhdWrrJWZHHSjANBgkqhkiG9w0BAQUFADBh
MQswGQYDVQQLFjVpZEVuMRMCA1UEFChMMRBGlnalUNlcnQoSW5iMRkwEwYDVQQLFy
Submit

5. Specify a unique name and make sure the certificate is **Shared**, and click **Submit**.

Create an Authorization Profile

Authorization Profile can be created at the Service Manager level or the deployment level. If you create an Authorization Profile in the Service Manager, then it can be shared by multiple deployments. If you create an Authorization Profile within a deployment, then it is only available for use within that deployment.

Use the following steps to set up an authorization profile for a deployment:

1. From the Service Manager, click **Service Manager** and then select Authorization Profile to open the Authorization Profiles page. To add an Authorization Profile for a specific deployment, click Deployment and select the deployment name from the left-navigation pane. Then select the Authorization Profile option.
2. On the Authorization Profiles page, click the plus sign (+) next to **Authorization Profiles**. The Authorization Profiles dialog box is displayed.
3. In the Authorization Profile dialog box, you can select from one of the external IdP providers and configure the options.

If you select the IDCS authorization profile, you need to configure the options shown in the following image:

Note

Configuring IDCS, IAM, or Microsoft Entra ID as external IdPs require similar values to be specified.

Add Authorization Profile ✕

GoldenGate authorization delegated to an external Identity provider (IdP). Make sure that the Security Role field in the Group Mapping section is filled.

Type: IDCS ▼

Enable Profile ☐

IDCS
IAM
OAM
Microsoft Entra ID

Description

Client ID Required

Client Secret Required

Group Mapping

Security Role Required

Administrator Role

Operator Role

User Role

Submit

The options that need to be configured are based on the configurations done on the external identity provider. Keep those values ready before you bring up this configuration. Refer to these values from [Configure IAM for Oracle GoldenGate](#).

These values are described as follows:

- Profile Name: Name of the authorization profile.
- Description (optional): Short summary of the profile being created.
- Enable Profile: Activates the profile for the deployment.
- Authorization Profile Type: IDCS
- Tenant Discovery URI: IDP server's OpenID Discovery Docs endpoint (/ .well-known/ openid-configuration).
- Client ID: IDP application's client ID
- Client Secret: IDP application's client secret (securely stored)
- In the **Group Mapping** section, the user mapping for IDCS groups to Oracle GoldenGate user roles is configured. You need to enter the name of the IDCS group with the corresponding user role. These values are case-sensitive. The user role options that map the name of a group with respective role in IDCS include **Security**, **Administrator**, **Operator**, and **User**.

4. If you select OAM as the external IdP, the following options are displayed:

Add Authorization Profile

×

Access to GoldenGate can be managed with an external Identity Provider (IdP). The Profile details and at least the Security Role for the Group Mapping.

Type

OAM

Profile Name

Required

Tenant Discovery URI

Required

Client ID

Required

Group Mapping

Security Role

Required

Administrator Role

Operator Role

User Role

Description

Identity Domain

Required

Client Secret

Required

Submit

To know the required values, see [Configure OAM for Oracle GoldenGate](#).

- Profile Name: Name of the authorization profile.
- Description (optional): Short summary of the profile being created.

Microservices Architecture Documentation
G49008-02
Copyright © 2017, 2026, Oracle and/or its affiliates.

January 22, 2026
Page 29 of 38

- **Enable Profile:** Activates the profile for the deployment.
- **Authorization Profile Type:** IDCS
- **Tenant Discovery URI:** IDP server's OpenID Discovery Docs endpoint (/.well-known/openid-configuration).
- **Identity Domain:** An Identity Domain corresponds to the notion of a tenant. All clients and resource servers are created under an Identity Domain.
- **Client ID:** IDP application's client ID
- **Client Secret:** IDP application's client secret (securely stored)
- In the **Group Mapping** section, the user mapping for IDCS groups to Oracle GoldenGate user roles is configured. You need to enter the name of the IDCS group with the corresponding user role. These values are case-sensitive. The user role options that map the name of a group with respective role in IDCS include **Security Administrator**, **Operator**, and **User**.

For more information, see [Creating an Identity Domain](#).

5. If you select EntraID, then the following options are displayed:

Add Authorization Profile

GoldenGate authorization delegated to an external Identity provider (iDP). Make sure that the Security Role field in the Group Mapping section is filled.

Type: Microsoft Entra ID

Enable Profile: ☐

Profile Name: EntraIDTest

Description:

Tenant Discovery URI: https://login.microsoftonline.com/

Client ID: 8788

Client Secret:

EntraID Values

Group Mapping

Security Role: EntraIDTest GG Security

Administrator Role: EntraIDTest GG Admin

Operator Role: EntraIDTest GG Operator

User Role: EntraID Test GG User

EntraID Group Mapping to Oracle GoldenGate Roles

Submit

To know the required values, see [Configure Microsoft Azure EntraID for Oracle GoldenGate](#).

6. Click **Submit** to create an authorization profile.
7. See the steps in [Test the Authorization Profile](#) to validate that the profile is correctly configured.
8. After you validate the profile successfully, you can enable the authorization profile.

Test the Authorization Profile

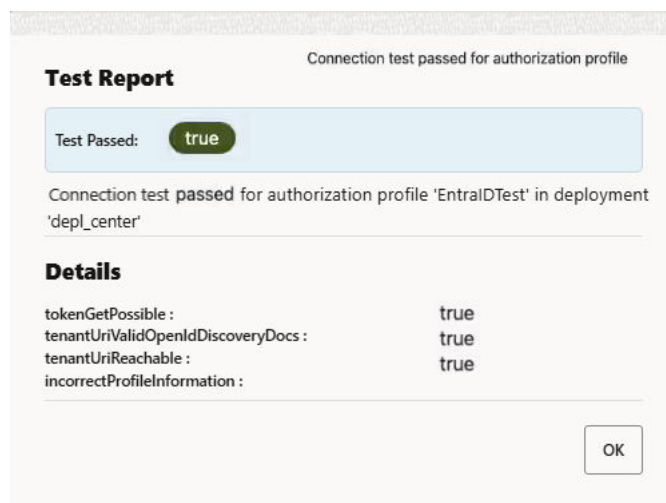
You should test the authorization profile that you have created before you enable it for usage. In this section, you'll test the Microsoft EntraID authorization profile. When the authorization profile is successfully set up, you will enable it from the Oracle GoldenGate Authroization Profile page in the Service Manager.

The following steps enable you to validate your authorization profile and enable it:

1. Log in to Service Manager and navigate to the Authorization Profile page. The Microsoft EntraID authorization profile is listed on this page.

Name	Type	Enabled	Client ID	Tenant Discovery URL	Identity Domain	Group Mapping	Description	Actions
localCredentialStore	localCredentialStore		N/A	N/A	N/A	N/A	Local Credential Store	
EntraIDTest	Microsoft Entra ID		d57678c-a2a8-4e04-9b22-ef3d5c155db	https://login.microsoftonline.com/.../oauth2/authorize	N/A			

2. Click the Validate profile icon. After the profile is validated, you will see a dialog box similar to the following:



3. Click the pencil icon in the Action column of the authorization profile and select Enable the authorization profile.

Edit Microsoft Entra ID Authorization Profile

GoldenGate authorization delegated to an external Identity provider (IdP). Make sure that the Security Role field in the Group Mapping section is filled.

Type: Microsoft Entra ID

Enable Profile ☒

Profile Name: EntraIDTest

Description:

Tenant Discovery URI: <https://login.microsoftonline.com/5b743bc7-c1e2-4d46-b4b5-a32>

Client ID: d6767f3c-a2a8-4e04-9b22-efa3a5c155db

Client Secret:

Group Mapping

Security Role: EntraID Test GG Security

Administrator Role: EntraID Test GG Admin

Operator Role: EntraID Test GG Operator

User Role: EntraID Test GG User

Submit

The Enabled column displays the status of the authorization profile.

Authorization Profiles +									
Name	Type	Enabled	Client ID	Tenant Discovery URI	Identity Domain	Group Mapping	Description	Actions	
localCredentialStore	localCredentialStore		N/A	N/A	N/A	N/A	Local Credential Store		
EntraIDTest	Microsoft Entra ID		d6767f3c-a2a8-4e04-9b22-efa3a5c155db	https://login.microsoftonline.com/5b743bc7-c1e2-4d46-b4b5-a32	N/A				

Switch Between States for a Deployment and Microservices

Learn about managing the status of the deployment and the Microservices.

Change the State of a Deployment

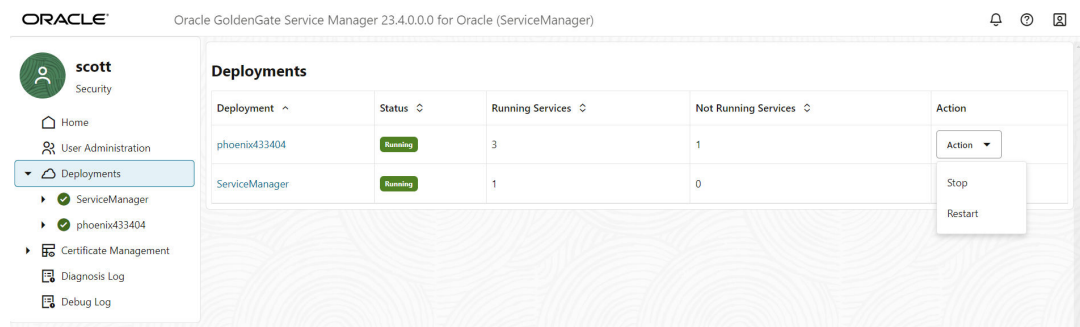
The state of a deployment is visible from the **Status** column of the **Deployments** section of the Service Manager home page. It is either in **Running** or **Stopped** state.

Note

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

To change the state of a deployment:

1. Log in to the Service Manager using the administrator account credentials.
2. In the **Deployments** section, click the **Action** button for the deployment that you need to start or stop.



3. Select from the available options for the deployment state change:
 - **Start:** If the deployment is in stopped state, this option allows you to start the deployment.
 - **Stop:** If the deployment is running, this option allows you stop it.
 - **Restart:** If the deployment is running but there are certain changes that are applied upon restart, then this option allows you to restart the deployment.

The option displayed depends on the current state of the deployment.

4. Verify that all the microservices associated with the deployment are in the same state as the deployment. By default, all microservices are in **Running** state after the deployment process is successful.

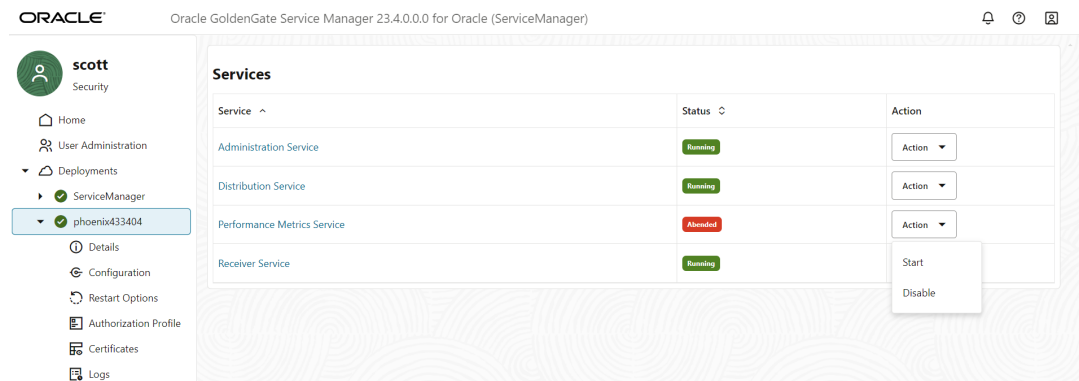
Change the State of Microservices in a Deployment

You can toggle between the states of microservices, to manage errors or apply changes to a deployment configuration in microservices. The microservices can be in the following states:

- Running
- Stopped
- Disabled

To change the state of the microservices associated with a deployment:

1. From the Service Manager home page, click the Deployment name from the left navigation pane. The Microservices for the deployment show in the right pane.
2. Click the **Action** button for the microservice for which the state needs to be changed.



3. Choose from the available options to change the state of the microservice:
 - **Start/Stop:** If the microservice is running, then the **Stop** option is available, and if its stopped, then the **Start** option appears.
 - **Disable/Enable:** If the microservice is in **Stopped** state, only then you can use the **Disable** option to disable the service. When a microservice is disabled, then the **Action** button changes to the **Enable** button, which implies that to change the state of the microservice, you would first need to enable it.

Access Configuration and Log Details from the Service Manager and Microservices

In the left-pane of the Service Manager web page, the Deployment section has two sub-sections, Service Manager and deployment. Both these sub-sections contain the options, which allow configuring the Service Manager or deployment.

The following images show the configuration options for Service Manager and the deployment.

Service Manger Configuration Options

ORACLE®

Oracle GoldenGate Service

scott

Security

Home

User Administration

Deployments

ServiceManager

Details

Configuration

Restart Options

Authorization Profile

Certificates

Logs

phoenix433404

Deployments

Deployment ^

phoenix433404

ServiceManager

Service Manager Configuration Options

Deployment Configuration Options

ORACLE®

Oracle GoldenGate Service Man

scott

Security

Home

User Administration

Deployments

ServiceManager

phoenix433404

Details

Configuration

Restart Options

Authorization Profile

Certificates

Logs

Services

Service ^

Administration Service

Distribution Service

Performance Metrics Service

Receiver Service

Deployment Configuration Options

Click the Service Manager option to expand the Service Manager configuration options.

Click the deployment name to expand the deployment configuration options.

The following table describes the configuration options for Service Manager and deployment.

Configuration Option	Service Manager	Deployment
Details	Click the Details option under Service Manager. It displays information regarding the following: • Status: Displays if the Service Manager is in Running state. • GoldenGate Home: Displays the location of the Oracle GoldenGate software home directory. This location can be modified or upgraded. • GoldenGate Etc Home: Displays the location of the /etc home directory. • GoldenGate Config Home: Displays the location of the configuration home directory. • GoldenGate SSL Home: Displays the SSL home directory location, which is under the /etc directory. • GoldenGate Var Home: Displays the /var home directory location. • GoldenGate Data Home: Displays the location of the data store for the configured deployment. • Managed by Service Manager: Confirms that the Oracle GoldenGate Microservices software installation is managed by the Service Manager.	Click the Details option under deployment name. It displays information regarding the following: • Status: Displays if the Service Manager is in Running state. • GoldenGate Home: Displays the location of the Oracle GoldenGate software home directory. This location can be modified or upgraded. • GoldenGate Etc Home: Displays the location of the /etc home directory. • GoldenGate Config Home: Displays the location of the configuration home directory. • GoldenGate SSL Home: Displays the SSL home directory location, which is under the /etc directory. • GoldenGate Var Home: Displays the /var home directory location. • GoldenGate Data Home: Displays the location of the data store for the configured deployment. • GoldenGate Archive Home: This is the location where the archived data files are stored. • Managed by Service Manager: Confirms that the Oracle GoldenGate Microservices software installation is managed by the Service Manager.
Configuration	Displays the list of environment variables configured for the Service Manager and deployment. You can add, edit, or delete environment variables from this page.	Displays the list of environment variables configured for the Service Manager and deployment. You can add, edit, or delete environment variables from this page.

Configuration Option	Service Manager	Deployment
Restart Options	Click this option to view the restart parameters for the Service Manager. The restart options for the Service Manager cannot be edited. The table in the right-pane shows the following values: • Enabled: Sets the restart of the service if the service terminates. Enabled by default. • On Success: Displays the value 'true' when Service Manager is running or started successfully. • Disable on Failure: Sets that the task is disabled when all retry attempts fail in the specified window. Enabled by default. • Auto Restart: When Service Manager fails to restart, the process will attempt to restart 9 times within a time frame of 1 minute(s). Each retry attempt will be separated by 0 minute(s).	See Manage Auto Start and Auto Restart for Extract and Replicat Processes .
Authorization Profile	See Create an Authorization Profile	See Create an Authorization Profile
Certificates	See Manage Certificates for Deployments	See Manage Certificates for Deployments
Logs	Displays the log activity for the Service Manager.	Displays the log activity for the Oracle GoldenGate Microservices. You can select the service from the drop down list to view the activities associated with a particular service.

Manage the Debug Log

Learn about managing the Debug Log.

Enable 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** toggle switch to start logging debug information.

Use the Debug Log

You can view, download, delete the debug log file to from this page. It is recommended that you delete the debug log after some time. You can maintain a local copy of the debug log to

help with debugging issues and then delete the debug log to avoid space issues on the host server.

1. Click the **Download Debug 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. Click the **Delete Debug Log File** button to delete a debug log.
4. Search for specific entries in the debug log using the **Search By** box, if required.
5. Click **Refresh** to get the latest log information, if it doesn't get refreshed automatically.

7

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 Databases

Learn about preparing and configuring Oracle and non-Oracle databases for Oracle GoldenGate.

Db2 LUW

With Oracle GoldenGate for Db2 LUW, you can perform initial loads and capture transactional data from supported Db2 LUW database versions and replicate the data to a Db2 LUW database or other supported Oracle GoldenGate targets, such as an Oracle Database.

Oracle GoldenGate for Db2 LUW supports data filtering, mapping, and transformations unless noted otherwise in this documentation.

Prepare Database Users and Privileges for Db2 LUW

Learn about creating database users and assigning privileges for Oracle GoldenGate for Db2 LUW.

- Create a database user that is dedicated to Oracle GoldenGate. It can be the same user for all of the Oracle GoldenGate processes that must connect to a database:
 - Extract (source database)
 - Replicat (target database)
 - DEFGEN (source or target database)
- 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, the Oracle GoldenGate database user. It is recommended that you store the login credentials in an Oracle GoldenGate credential store. The credential store makes use of local secure storage for the login names and passwords, and permits you to specify only an alias in the Oracle GoldenGate parameter files.
- Assign system administrator (SYSADM) or database administrator (DBADM) authority to the database user under which Extract runs. To give the Extract user DBADM authority, a user with SYSADM authority can issue the following grant statement.

```
GRANT DBADM ON DATABASE TO USER user
```

This authority can also be granted from the User and Group Objects folder in the DB2 Control Center. The database tab for the user that is assigned to an Oracle GoldenGate process should have the Database Administrative Authority box checked.

Note

If the Extract user does not have the required authority, Extract will log the following errors and stop.

```
[SC=-1224:SQL1224N A database agent could not be started to  
service a request, or was terminated as a result of a database  
system shutdown or a force command.  
SQL STATE 55032: The CONNECT statement is invalid, because the  
database manager was stopped after this application was started]
```

- Grant at least the following privileges to the database user under which Replicat runs:
 - Local CONNECT to the target database
 - SELECT on the system catalog views
 - SELECT, INSERT, UPDATE, and DELETE on the target tables

Prepare Database Connection, System, and Parameter Settings

Learn about configuring database connections for Oracle GoldenGate for Db2 LUW.

To set up the database connection from Oracle GoldenGate for a Db2 LUW deployment, see [Add Database Connections](#).

Prepare Tables for Processing

You must perform the following tasks to prepare your tables for use in an Oracle GoldenGate environment for Db2 LUW.

Disabling Triggers and Cascade Constraints

Disable triggers, cascade delete constraints, and cascade update constraints on target Db2 LUW tables. Oracle GoldenGate replicates DML that results from a trigger or cascade constraint. If the same trigger or constraint gets activated on the target table, it becomes redundant because of the replicated version, and the database returns an error. Consider the following example, where the source tables are emp_src and salary_src and the target tables are emp_targ and salary_targ.

1. A delete is issued for emp_src.
2. It cascades a delete to salary_src.
3. Oracle GoldenGate sends both deletes to the target.
4. The parent delete arrives first and is applied to emp_targ.
5. The parent delete cascades a delete to salary_targ.
6. The cascaded delete from salary_src is applied to salary_targ.
7. The row cannot be located because it was already deleted in step 5.

Ensuring Row Uniqueness for Tables

Oracle GoldenGate requires some form of 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:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. 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 those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration.

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.

4. If a table does not have an appropriate key, or if you prefer that the existing key(s) are not 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. See `TABLE | MAP` in *Parameters and Functions Reference for Oracle GoldenGate*.

Using `KEYCOLS` to Specify a Custom Key

If a table does not have one of the preceding types of row identifiers, or if you prefer those identifiers 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.

Preventing Key Changes

Do not add columns to a key after Oracle GoldenGate starts extracting data from the table. This rule applies to a primary key, a unique key, a `KEYCOLS` key, or an all-column key. Db2 LUW does not supply a before image for columns that are added to a table. If any columns in a key are updated on the source, Oracle GoldenGate needs a before image to compare with the current values in the target table when it replicates the update.

Maintaining Materialized Query Tables

To maintain parity between source and target materialized query tables (MQT), you replicate the base tables, but not the MQTs. The target database maintains the MQTs based on the changes that Replicat applies to the base tables.

The following are the rules for configuring these tables:

- Include the base tables in your `TABLE` and `MAP` statements.
- Do not include MQTs in the `TABLE` and `MAP` statements.

- Wildcards can be used in TABLE and MAP statements, even though they might resolve MQT names along with regular table names. Oracle GoldenGate automatically excludes MQTs from wildcarded table lists. However, any MQT that is explicitly listed in an Extract TABLE statement by name will cause Extract to abend.

Creating a Temporal Table

A temporal table is a table that maintains the history of its data and the time period when its data are valid. Temporal tables are used in Oracle GoldenGate to keep track of all the old rows that are deleted or updated in the table. Temporal tables are also used to maintain the business validity of its rows and data. For example, Oracle GoldenGate keeps track of the time period during which a row is valid. There are three types of temporal tables, system-period, application-period, and bitemporal table.

Convert to a Temporal Table

You can convert an already existing table into a temporal table, which changes the structure of the table. This section describes how the structure of the tables changes. The following sample existing table is converted into all three temporal tables types in the examples in this section:

Table policy_info

```
(
Policy_id char[4] not null primary key,
Coverage int not null
)
```

And the tables contains the following initial rows

POLICY_ID	COVERAGE
ABC	12000
DEF	13000
ERT	14000

Example 1 Converting an existing table into System-period temporal table.

You convert the sample existing table into a system-period temporal table by adding SYSTEM_PERIOD, transaction id columns, and SYSTEM_TIME period as in the following:

```
ALTER TABLE policy_info
  ADD COLUMN sys_start TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN;
ALTER TABLE policy_info
  ADD COLUMN sys_end TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END;
ALTER TABLE policy_info
  ADD COLUMN ts_id TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID;
ALTER TABLE policy_info ADD PERIOD SYSTEM_TIME(sys_start, sys_end);
```

Then you create a history table for the new temporal table using one of the following two methods:

- ```
CREATE TABLE hist_policy_info
(
 policy_id CHAR(4) NOT NULL,
 coverage INT NOT NULL,
 sys_start TIMESTAMP(12) NOT NULL ,
 sys_end TIMESTAMP(12) NOT NULL,
 ts_id TIMESTAMP(12) NOT NULL
);
ALTER TABLE hist_policy_info ADD RESTRICT ON DROP;
```

- `CREATE TABLE hist_policy_info LIKE policy_info with RESTRICT ON DROP;`

The `RESTRICT ON DROP` clause will not allow the history table to get dropped while dropping system-period temporal table. Otherwise the history table gets implicitly dropped while dropping its associated temporal table. You can create a history table without `RESTRICT ON DROP`. A history table cannot be explicitly dropped.

You should not use the `GENERATED ALWAYS` clause while creating a history table. The primary key of the system-period temporal table also does not apply here as there could be many updates for a particular row in the base table, which triggers many inserts into the history table for the same set of primary keys. Apart from these, the structure of a history table should be exactly same as its associated system-period temporal table. The history table must have the same number and order of columns as system-period temporal table. History table columns cannot explicitly be added, dropped, or changed. You must associate a system-period temporal table with its history table with the following statement:

```
ALTER TABLE policy_info ADD VERSIONING USE HISTORY TABLE hist_policy_info.
```

The `GENERATED ALWAYS` columns of the table are the ones that are always populated by the database manager so you do not have any control over these columns. The database manager populates these columns based on the system time.

The extra added `SYSTEM_PERIOD` and `transaction id` columns will have default values for already existing rows as in the following:

| POLICY_ID                        | COVERAGE                               | TS_ID |
|----------------------------------|----------------------------------------|-------|
| -----                            | -----                                  | ----- |
| -----                            | -----                                  | ----- |
| ABC                              | 12000 0001-01-01-00.00.00.000000000000 |       |
| 9999-12-30-00.00.00.000000000000 | 0001-01-01-00.00.00.000000000000       |       |
| DEF                              | 13000 0001-01-01-00.00.00.000000000000 |       |
| 9999-12-30-00.00.00.000000000000 | 0001-01-01-00.00.00.000000000000       |       |
| ERT                              | 14000 0001-01-01-00.00.00.000000000000 |       |
| 9999-12-30-00.00.00.000000000000 | 0001-01-01-00.00.00.000000000000       |       |

The associated history table is populated with the before images once you start updating the temporal table.

## Example 2 Converting an existing table into application-period temporal table.

You can convert the sample existing table into application-period temporal table by adding time columns and a `BUSINESS_TIME` period as in the following:

```
ALTER TABLE policy_info ADD COLUMN bus_start DATE NOT NULL DEFAULT '10/10/2001'
ALTER TABLE policy_info ADD COLUMN bus_end DATE NOT NULL DEFAULT '10/10/2002'
ALTER TABLE policy_info ADD PERIOD BUSINESS_TIME(bus_start, bus_end)
```

While adding time columns, you need to make sure that while entering business validity time values of the existing time columns, the `bus_start` column always has value lesser than `bus_end` because these columns specify the business validity of the rows.

The new application-period temporal table will look similar to:

| POLICY_ID | COVERAGE | BUS_START  | BUS_END    |
|-----------|----------|------------|------------|
| -----     | -----    | -----      | -----      |
| ERT       | 14000    | 10/10/2001 | 10/10/2002 |
| DEF       | 13000    | 10/10/2001 | 10/10/2002 |
| ABC       | 12000    | 10/10/2001 | 10/10/2002 |

**Example 3 Converting an existing table into bitemporal table.**

You can convert the sample existing table into bitemporal table by adding SYSTEM\_PERIOD, time columns along with the SYSTEM\_TIME and BUSINESS\_TIME period as in the following:

```
ALTER TABLE policy_info
 ADD COLUMN sys_start TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN;
ALTER TABLE policy_info
 ADD COLUMN sys_end TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END;
ALTER TABLE policy_info
 ADD COLUMN ts_id TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID;
ALTER TABLE policy_info ADD PERIOD SYSTEM_TIME(sys_start, sys_end);

ALTER TABLE policy_info ADD COLUMN bus_start DATE NOT NULL DEFAULT '10/10/2001'
ALTER TABLE policy_info ADD COLUMN bus_end DATE NOT NULL DEFAULT '10/10/2002'
ALTER TABLE policy_info ADD PERIOD BUSINESS_TIME(bus_start, bus_end)
```

While adding the time columns, you must make sure that while entering business validity time values of already existing time columns, the bus\_start column always has value lesser than bus\_end because these columns specify the business validity of the rows.

Then you create a history table for the new temporal table using one of the following two methods:

- ```
CREATE TABLE hist_policy_info
(
  policy_id    CHAR(4) NOT NULL,
  coverage     INT NOT NULL,
  sys_start    TIMESTAMP(12) NOT NULL ,
  sys_end      TIMESTAMP(12) NOT NULL,
  ts_id        TIMESTAMP(12) NOT NULL
);
ALTER TABLE hist_policy_info ADD RESTRICT ON DROP;
CREATE TABLE hist_policy_info LIKE policy_info with RESTRICT ON DROP;
```
- The RESTRICT ON DROP clause will not allow the history table to get dropped while dropping system-period temporal table. Otherwise the history table gets implicitly dropped while dropping its associated temporal table. You can create a history table without RESTRICT ON DROP. A history table cannot be explicitly dropped.

You should not use the GENERATED ALWAYS clause while creating a history table. The primary key of the system-period temporal table also does not apply here as there could be many updates for a particular row in the base table, which triggers many inserts into the history table for the same set of primary keys. Apart from these, the structure of a history table should be exactly same as its associated system-period temporal table. The history table must have the same number and order of columns as system-period temporal table. History table columns cannot explicitly be added, dropped, or changed. You must associate a system-period temporal table with its history table with the following statement:

```
ALTER TABLE policy_info ADD VERSIONING USE HISTORY TABLE hist_policy_info.
```

The GENERATED ALWAYS columns of the table are the ones that are always populated by the database manager so you do not have any control over these columns. The database manager populates these columns based on the system time.

The extra added SYSTEM_PERIOD and transaction id columns will have default values for already existing rows as in the following:

POLICY_ID	COVERAGE
SYS_START	
SYS_END	TS_ID
-----	-----
-----	-----
ABC	12000 0001-01-01-00.00.00.000000000000
9999-12-30-00.00.00.000000000000	0001-01-01-00.00.00.000000000000
DEF	13000 0001-01-01-00.00.00.000000000000
9999-12-30-00.00.00.000000000000	0001-01-01-00.00.00.000000000000
ERT	14000 0001-01-01-00.00.00.000000000000
9999-12-30-00.00.00.000000000000	0001-01-01-00.00.00.000000000000

The associated history table is populated with the before images once you start updating the temporal table.

The extra added SYSTEM_TIME period, transaction id and time columns will have default values for already existing rows as in the following:

POLICY_ID	COVERAGE	SYS_START	SYS_END
TS_ID		BUS_START	BUS_END
-----	-----	-----	-----
-----	-----	-----	-----
ABC	12000 0001-01-01-00.00.00.000000000000	9999-12-30-00.00.00.000000000000	
0001-01-01-00.00.00.000000000000	10/10/2001	10/10/2002	
DEF	13000 0001-01-01-00.00.00.000000000000	9999-12-30-00.00.00.000000000000	
0001-01-01-00.00.00.000000000000	10/10/2001	10/10/2002	
ERT	14000 0001-01-01-00.00.00.000000000000	9999-12-30-00.00.00.000000000000	
0001-01-01-00.00.00.000000000000	10/10/2001	10/10/2002	

The history table is populated with the before images once user starts updating the temporal table.

Example 4 Replication in Heterogeneous Environment.

In heterogeneous configuration in which you do not have temporal tables at the apply side, you can only replicate the system-period and bitemporal tables though *not* the associated history tables. While performing replication in this situation, you must take care of the SYSTEM_PERIOD and transaction id columns value. These columns will have some values that the target database might not support. You should first use the map conversion functions to convert these values into the format that the target database supports, and then map the columns accordingly.

For example, MySQL has a DATETIME range from 1000-01-01 00:00:00.000000 to 9999-12-31 23:59:59.999999. You cannot replicate a timestamp value of 0001-01-01-00.00.00.000000000000 to MySQL. To replicate such values, you must convert this value into the MySQL DATETIME value 1000-01-01 00:00:00.000000, and then map the columns. If you have the following row in the policy_info system-period table:

POLICY_ID	COVERAGE
SYS_START	
SYS_END	TS_ID
-----	-----
-----	-----
ABC	12000 0001-01-01-00.00.00.000000000000
0001-01-01-00.00.00.000000000000	9999-12-30-00.00.00.000000000000

To replicate the row into MySQL, you would use the colmap() function:

```
map source_schema.policy_info, target target_schema.policy_info colmap
(policy_id=policy_id, coverage=coverage, sys_start= @IF( ( @NUMSTR( @STREXT(sys_
```

```
start,1,4))) > 1000, sys_start, '1000-01-01 00.00.00.000000'), sys_end=sys_end,
ts_id= @IF( ( @NUMSTR( @STREXT(ts_id,1,4))) > 1000, ts_id, '1000-01-01
00.00.00.000000')));
```

Replicating with Temporal Tables

You can choose one of the following methods to replicate a system-period or a bitemporal temporal table as follows:

- You can replicate a temporal table to another temporal table only; this is the default behavior. Oracle GoldenGate will not replicate the `SYSTEM_TIME` period and transaction id columns because these are automatically generated columns at the apply side. The database manager populates the columns in the target temporal table using the system clock time and with the default values. You can preserve the original values these columns then use any of the following:
 - Add extra timestamp columns in the target temporal table and map the columns accordingly. The extra columns are automatically added to the associated history table.
 - Use a non-temporal table at the apply side and map the columns appropriately. In this scenario, you will not be able to maintain the history table.
 - In a heterogeneous configuration where the source is Db2 LUW and the target is a different database, you can either ignore the automatically generated columns or use an appropriate column conversion function to convert the columns value in the format that target database supports and map them to target columns accordingly.

Or

- You can replicate a temporal table, with the associated history table, to a temporal and history table respectively then you must specify the replicate parameter, `DBOPTIONS SUPPRESSTEMPORALUPDATES`. You must specify both the temporal table and history table to be captured in the Extract parameter file. Oracle GoldenGate replicates the `SYSTEM_TIME` period and transactions id columns value. You must ensure that the database instance has the execute permission to run the stored procedure at the apply side.

Oracle GoldenGate cannot detect and resolve conflicts while using default replication as `SYSTEM_TIME` period and `transactionstart id` columns remains auto generated. These columns cannot be specified in set and where clause. If you use the `SUPPRESSTEMPORALUPDATES` parameter, then Oracle GoldenGate supports CDR.

Configuring the Transaction Logs for Oracle GoldenGate

To capture DML operations, Oracle GoldenGate reads the Db2 LUW online logs by default. However, it reads the archived logs if an online log is not available. To ensure the continuity and integrity of Oracle GoldenGate processing, configure the logs as follows.

Retaining the Transaction Logs

Configure the database to retain the transaction logs for roll forward recovery by enabling one of the following parameter sets, depending on the database version.

- Db2 LUW 10.1 and later:

Set the `LOGARCHMETH` parameters as follows:

 - Set `LOGARCHMETH1` to `LOGRETAIN`.
 - Set `LOGARCHMETH2` to `OFF`.

Alternatively, you can use any other `LOGARCHMETH` options, as long as forward recovery is enabled. For example, the following is valid:

- Set LOGARCHMETH1 to DISK.
- Set LOGARCHMETH2 to TSM.

To determine the log retention parameters:

1. Connect to the database.

db2 connect to *database* user *username* using *password*

2. Get the database name.

db2 list db directory

3. Get the database configuration for the database.

db2 get db cfg for *database*

The fields to view are:

Log retain for recovery status = RECOVERY
User exit for logging status = YES

To set the log retention parameters:

1. Issue one of the following commands.

To enable USEREXIT:

db2 update db cfg for *database* using USEREXIT ON

If not using USEREXIT, use this command:

db2 update db cfg for *database* using LOGRETAIN RECOVERY

To set LOGARCHMETH:

db2 update db cfg for *database* using LOGARCHMETH1 LOGRETAIN

db2 update db cfg for *database* using LOGARCHMETH2 OFF

2. Make a full backup of the database by issuing the following command.

db2 backup db *database* to *device*

3. Place the backup in a directory to which Db2 LUW has access rights. If you get the following message, contact your systems administrator:

SQL2061N An attempt to access media "device" is denied.

Specifying the Archive Path

Set the Db2 LUW OVERFLOWLOGPATH parameter to the archive log directory. The node attaches automatically to the path variable that you specify.

db2 connect to *database*

db2 update db cfg using overflowlogpath "*path*"

Exclude the node itself from the path. For example, if the full path to the archive log directory is /sdb2logarch/oltpods1/archive/OLTPODS1/NODE0000, then the OVERFLOWLOGPATH value should be specified as /sdb2logarch/oltpods1/archive/OLTPODS1.

Db2 LUW: Supported Data Types, Objects, and Operations

This section contains information on supported data types, objects, and operations for Oracle GoldenGate on Db2 LUW.

Supported Db2 LUW Data Types

Oracle GoldenGate supports all Db2 LUW data types, except those listed in [Non-Supported Db2 LUW Data Types](#).

Limitations of Support

Oracle GoldenGate has the following limitations for supporting Db2 LUW data types:

- Oracle GoldenGate supports multi-byte character data types and multi-byte data stored in character columns. Multi-byte data is only supported in a like-to-like configuration. Transformation, filtering, and other types of manipulation are not supported for multi-byte character data.
- BLOB and CLOB columns must have a LOGGED clause in their definitions.
- Due to limitations in the IBM DB2READLOG interface, Oracle GoldenGate does not support coordination of transactions across nodes in a Db2 Database Partitioning Feature (DPF) environment. In DPF, a transaction may span multiple nodes, depending upon how the data is partitioned.

However, if you need to capture from it, you can do it with certain limitations. Check the Oracle Support note [Does Oracle GoldenGate Support DB2 LUW Data Partitioning Feature \(DPF\)?](#) (DocID 2763006.1).

- GRAPHIC and VARGRAPHIC columns must be in a database, where the character set is UTF16. Any other character set causes the Oracle GoldenGate toabend.
- The support of 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.
- Extract fully supports the capture and apply of TIMESTAMP(0) through TIMESTAMP(12) when the output trail format is 19.1 or higher. Otherwise Extract truncates data from TIMESTAMP(10) through TIMESTAMP(12) to nanoseconds (maximum of nine digits of fractional time) and issues a warning to the report file.
- 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.
- Oracle GoldenGate does not support the filtering, column mapping, or manipulation of large objects that are larger than 4K. You can use the full Oracle GoldenGate functionality for objects that are 4K or smaller.
- Replication of XML columns between source and target databases with the *same* character set is supported. If the source and target database character sets are different, then XML replication may fail with a database error because some characters may not be recognized (or valid) in the target database character set.
- Oracle GoldenGate does not certify capturing changes from the Db2 LUW column-organized tables.

- The DECFLOAT data type supports the NaN, sNaN, Infinity/-Infinity special numeric values.

For more information on data types by databases that support special values, refer to [Special Values Supported by Database Data Types](#).

Non-Supported Db2 LUW Data Types

The non-supported Db2 LUW data types are:

- XMLType (Capture)
- User-defined types
- Negative dates

Supported Objects and Operations for Db2 LUW

Object and operations that are supported for Db2 LUW are:

- Oracle GoldenGate Extract supports cross-endian capture where the database and Oracle GoldenGate are running on different byte order servers. The byte order is detected automatically for Db2 LUW version 10.5 or higher. If the Db2 database auto-detection on the Db2 LUW 10.5 database is not required then you can override it by specifying the `TRANLOGOPTIONS MIXEDENDIAN [ON|OFF]` parameter. For Db2 LUW version 10.1, this parameter must be used in the Extract parameter file for cross-endian capture. See `TRANLOGOPTIONS` for more information.
- Db2 pureScale environment is supported.
- Oracle GoldenGate supports the maximum number of columns and column size per table that is supported by the database.
- `TRUNCATE TABLE`.
- Multi-Dimensional Clustered Tables (MDC).
- Materialized Query Tables. Oracle GoldenGate does not replicate the MQT itself, but only the base tables. The target database automatically maintains the content of the MQT based on the changes that are applied to the base tables by Replicat.
- Tables with `ROW COMPRESSION`. In Db2 LUW version 10.1 and later, `COMPRESS YES STATIC` is supported and `COMPRESS YES ADAPTIVE` are supported.
- Extended row size feature is enabled by default. It is supported with a workaround using `FETCHCOLS`. For any column values that are `VARCHAR` or `VARGRAPHIC` data types and are stored out of row in the database, you must fetch these extended rows by specifying these columns using the `FETCHCOLS` option in the `TABLE` parameter in the extract parameter file. With this option set, when the column values are out of row then Oracle GoldenGate will fetch its value. If the value is out of and `FETCHCOLS` is *not* specified then Extract will abend to prevent any data loss. If you do not want to use this feature, set the `extended_row_size` parameter to `DISABLE`.

Extended row size feature is enabled, by default. It is supported with a workaround using `FETCHCOLS` for Db2 LUW 10.1. For any column values that are `VARCHAR` or `VARGRAPHIC` data types and are stored out of row in the database, you must fetch these extended rows by specifying these columns using the `FETCHCOLS` option in the `TABLE` parameter in the Extract parameter file. With this option set, when the column values are out of row, then Oracle GoldenGate fetches its value. If the value is out of and `FETCHCOLS` is not specified then Extract abends to prevent any data loss. If you do not want to use this feature, set the `extended_row_size` parameter to `DISABLE`. For Db2 LUW 10.5 and higher out of row

values are captured seamlessly by Extract. FETCHCOLS is no more needed to capture out of row columns from these database versions.

- Temporal tables with Db2 LUW 10.1 FixPack 2 and greater are supported. This is the default for Replicat.
 - Replication between system-period temporal tables and application-period temporal tables is not supported.
 - Replication from a non-temporal table to a temporal table is not supported.
 - Replication of temporal tables with the INSERTALLRECORDS parameter is not supported.
 - Bidirectional replication is supported only with the default replication.
 - CDR in bidirectional replication is not supported.
 - CDR in application-period temporal tables is supported.
- Supported options with SEND EXTRACT SHOWTRANS || SKIPTRANS || FORCETRANS command are:

Supported options with SHOWTRANS command

```
SHOWTRANS [transaction_ID] [COUNT n]
[DURATION duration unit]
[TABULAR][FILE file_name] |
```

Options with SKIPTRANS and FORCETRANS.

```
SKIPTRANS transaction_ID
[FORCE] FORCETRANS transaction_ID [FORCE]
```

- Limitations on Automatic Heartbeat Table support are as follows:
 - [THREAD n] [DETAIL] is not supported.
 - Oracle GoldenGate heartbeat parameters frequency and purge frequency are accepted in seconds and days. However, the Db2 LUW task scheduler accepts its schedule only in cron format so the Oracle GoldenGate input value to cron format may result in some loss of accuracy. For example:

```
ADD HEARTBEATTABLE, FREQUENCY 150, PURGE_FREQUENCY 20
```

This example sets the FREQUENCY to 150 seconds, which is converted to the closest minute value of 2 minutes, so the heartbeat table is updated every 120 seconds instead of every 150 seconds. Setting PURGE_FREQUENCY to 20 means that the history table is purged at midnight on every 20th day.

- The following steps are necessary for the heartbeat scheduled tasks to run:

1. Set the DB2_ATS_ENABLE registry variable to db2set DB2_ATS_ENABLE=YES.

2. Create the SYSTOOLSPACE tablespace if it does not already exist:

```
CREATE TABLESPACE SYSTOOLSPACE IN IBMCATGROUP MANAGED BY AUTOMATIC STORAGE
EXTENTSIZE 4
```

3. Ensure instance owner has Database administration authority (DBADM):

```
GRANT DBADM ON DATABASE TO instance_owner_name
```

Non-Supported Objects and Operations for Db2 LUW

Objects and operations for Db2 LUW that are not supported by Oracle GoldenGate are:

- Schema, table or column names that have trailing spaces
- Multiple instances of a database
- Extraction or replication of DDL (data definition language) operations
- Generated columns (GENERATE ALWAYS clause)
- DB2 Data Partitioning Feature (DPF) is not supported. DPF doesn't provide a way to read the log records in a coordinated fashion across all the nodes in a partition. So, there is no way to ensure that even if all nodes are being replicated with separate Extracts, it would be possible to ensure that all records from all transactions are applied in the correct temporal order. This may occur due to a number of factors including caching in the database and the underlying operating system not allowing the Extracts to have visibility into whether there are any cached and not yet visible entries that may affect the ordering of the record operations across all partitions. Due to this uncertainty, it is not possible to ensure that transactions that span partitions, or primary key updates can be replicated in a consistent manner.

System Schemas

The following schemas or objects are not be automatically replicated by Oracle GoldenGate unless they are explicitly specified without a wildcard.

- "SYSIBM"
- "SYSCAT"
- "SYSSTAT"
- "SYSPROC"
- "SYSFUN"
- "SYSIBMADMIN"
- "SYSTOOLS"
- "SYSPUBLIC"

Supported Object Names

For a list of characters that are supported in object names, see [Supported Database Object Names](#).

Db2 for i

With Oracle GoldenGate for Db2 for i, you can perform initial loads and capture transactional data from supported Db2 for i versions and replicate the data to a Db2 for i database or other supported Oracle GoldenGate targets, such as an Oracle Database.

Oracle GoldenGate for Db2 for i supports data filtering, mapping, and transformations unless noted otherwise in this documentation.

Prepare Database Users and Privileges for Db2 for i

User Profiles and Security Privileges

The user who installs Oracle GoldenGate must have read and write privileges on the Oracle GoldenGate installation directory, as these privileges will be required later to perform steps to create sub-folders and run specific programs.

The objects in the Oracle GoldenGate library (specified with GGSHEMA), should have their ownership changed to the dedicated user profile or group profile for Oracle GoldenGate.

Dedicated User Profile Account

It is recommended that the Oracle GoldenGate processes on Db2 for i database be assigned a dedicated user or group profile, and is used by all Oracle GoldenGate processes. This user profile should not be used by any other application(s).

The dedicated user profile should be granted permission only to the objects that the Oracle GoldenGate will be operating on. If there is specific change data that is not to be accessed by Oracle GoldenGate processes, then such change data should not be included in the journals, which are accessed by Oracle GoldenGate and its dedicated user profile. All Oracle GoldenGate processes must have read, write, and delete object privileges within the Oracle GoldenGate installation library, as specified by GGSHEMA.

Security Privileges on a Db2 for i System

The Extract and Replicat user profiles need to be assigned the following authorities at a minimum:

- The simplest way to ensure Oracle GoldenGate will be able to operate is to assign *ALLOBJ authority to the Oracle GoldenGate user profile(s), however this is not necessary.
- The Manager process must have privileges to control all other Oracle GoldenGate processes (Db2 for i *JOBCTL authority).
- The Oracle GoldenGate user profiles(s) need at least the *USE authority to the *FILE objects in the QSYS2 library which contains the SQL catalog (which by default should be accessible to any user).
- Assign at least the *USE authority (*OBJOPR, *READ, *EXECUTE) to all the *FILE (table) and *JRNRCV (journal receiver) objects on the system that are accessed by the Extract user profile.
- Assign the following authorities to the *JRN (journal) objects that are accessed by the Extract user profile, in addition to the *USE authority (*OBJOPR, *READ, *EXECUTE): *OBJEXIST, *OBJREF, and *ADD.
- Assign the *CHANGE authority to all the *FILE objects on the system that are accessed by the Replicat user profile.

The Oracle GoldenGate user profile that runs the Extract process needs to have the *USE authority on the QSYS/QPMLPMGT service program.

These authorities must be granted through the native Db2 for i interface through a 5250 terminal session or through the Db2 for i Navigator product available from IBM.

Prepare Database Connection, System, and Parameter Settings

Learn about configuring database connection, system and parameters settings for Oracle GoldenGate for Db2 for i.

Enabling SSL

SSL connections can be enabled by setting SSL=1 in the DSN configuration file. To know about how to set up an SSL connection with IBM i Access ODBC Driver, see [Make SSL ODBC connections from Linux to Db2 for i](#) and [ACS ODBC driver for Linux now supports OpenSSL](#).

It is recommended to use OpenSSL to setup SSL. After SSL is enabled in the DSN configuration file, JAVA connections that have been established using jt400 libraries will also be using SecureAS400 connections as well.

Port Requirements for Db2 for i

Oracle GoldenGate for Db2 for i requires the following ports in normal and SSL modes:

- **Normal Mode:** 446, 449, 8470-8476
- **SSL Mode:** 448, 449, 9470-9476

Preparing the Journals for Data Capture by Extract

All tables for which you want data to be captured must be journaled, either explicitly or by default by means of a QSQJRN journal in the same library. To preserve data integrity, data journal entries are sent to the Extract process in time order as they appear on the system. This section provides guidelines for configuring the journals to support capture by the Extract process.

Allocating Journals to an Extract Group

One Extract process can process a single journal. If using more journals than that, use additional Extract processes to handle the extra journals. You can also use additional Extract processes to improve capture performance if necessary.

Note

To ensure transaction integrity, all objects that correspond to any given transaction must be read by the same Extract group.

Setting Journal Parameters

To support the capture of data by the Extract process, the following are the minimal journaling parameter settings that are required.

- Manage Receivers (MNGRCV): *SYSTEM
- Delete Receivers (DLTRCV): *NO
- Receiver Size Option (RCVSIZOPT): *MAXOPT2 (*MAXOPT3 recommended). Oracle GoldenGate fully supports either option.

- Journal State (JRNSTATE): *ACTIVE
- Minimize Entry Specific Data (MINENTDTA): *NONE
- Fixed Length Data (FIXLENTDTA): *USR

In the following example, the command to set these attributes for a journal JRN1 in library LIB1 would be:

```
CHGJRN JRN(LIB1/JRN1) MNGRCV(*SYSTEM) DLTRCV(*N0) RCVSIZOPT(*MAXOPT3)
JRNSTATE(*ACTIVE) MINENTDTA(*NONE) FIXLENTDTA(*USR)
```

Note

To check the attributes of a journal, use the command `WRKJRNA JRN(LIB1/JRN1) DETAIL(*CURATR)`.

When the journaling is set to the recommended parameter settings, you are assured that the entries in the journals contain all of the information necessary for Oracle GoldenGate processing to occur. These settings also ensure that the system does not delete the journal receivers automatically, but retains them in case Extract needs to process older data.

Deleting Old Journal Receivers

Although the DLTRCV parameter is set to N0 in the recommended configuration for Extract (see [Setting Journal Parameters](#)), you can delete old journal receivers manually once Extract is finished capturing from them.

If using another application that is using the journals that Oracle GoldenGate will be reading, consideration must be given regarding any automatic journal receiver cleanup that may be in place. Oracle GoldenGate must be able to read the journal receivers before they are detached or removed.

To Delete Journal Receivers

1. Run the following command to view the journal positions in which Extract has current processing points, along with their journal receivers.

```
INFO EXTRACT group
```

2. Use the following Db2 for i command to delete any journal receivers that were generated prior to the ones that are shown in the INFO EXTRACT command.

```
DLTJRNCV JRNCV(library/journal_receiver)
```

Where:

library and *journal_receiver* are the actual names of the library and journal receiver to be deleted. See the Db2 for i Information Center for more information about this command.

Using Remote Journal

This topic contains instructions for remote journal preparation and adding a remote journal. Remote Journal support in the IBM Db2 for i operating system provides the ability for a system to replicate, in its entirety, a sequence of journal entries from one Db2 for i system to another. Once setup, this replication is handled automatically and transparently by the operating

system. The entries that are replicated are placed in a journal on the target system that is available to be read by an application in the same way as on the source system.

You must have an understanding of how to setup and use remote journaling on an Db2 for i system to use this feature with Oracle GoldenGate. There are no special software requirements for either Oracle GoldenGate or the Db2 for i systems to use remote journaling.

Preparing to Use Remote Journals

Before establishing the remote journal environment, complete the following steps:

1. Determine the extent of your remote journal network or environment.
2. *Library redirection* is the ability to allow the remote journal and associated journal receivers to reside in different libraries on the target system from the corresponding source journal and its associated journal receivers.

Determine what library redirection, if any, you will be using for the remote journals and associated journal receivers.

3. Ensure that all selected libraries exist on the target systems. You must consider whether or not library redirection will be used when adding the remote journal.
4. Create the appropriate local journal if it does not already exist.
5. Configure and activate the communications protocol you have chosen to use.
6. After you have configured the communications protocol, it must be active while you are using the remote journal function.

For example, if you are using the OptiConnect for IBM i bus transport method, then the OptiConnect for IBM i subsystem, QSOC, must be active. QSOC must be active for both the source system and the target system, and the appropriate controllers and devices must be varied on. If you are using a SNA communications transport, vary on the appropriate line, controller, and devices and ensure subsystem QCMN is active on both systems. Start of change If you are using TCP/IP or Sockets IPv6, you must start TCP/IP by using the Start TCP/IP (STRTCP) command, including the distributed data management (DDM) servers. If you are using data port, you must configure a cluster, make sure that the cluster is active, and start the internet Daemon (inetd) server using the Start TCP/IP Server (STRTCPSVR) command.

7. If one does not already exist, create the appropriate relational database (RDB) directory entry that will be used to define the communications protocol for the remote journal environment. When TCP communications are being used to connect to an independent disk pool, the RDB entry to the independent disk pool must have the Relational database value set to the target system's local RDB entry and the relational database alias value set to the independent disk pool's name.
8. Now you should be able to see the remote database connection by issuing the WRKRDBDIRE command.

Work with Relational Database Directory Entries

Position to

Type options, press Enter.

1=Add 2=Change 4=Remove 5=Display details 6=Print details

Remote

Option Entry Location Text

SYS1 system1

SYS2 system2

MYSYSTEM *LOCAL Entry added by system

Bottom

F3=Exit F5=Refresh F6=Print list F12=Cancel F22=Display entire field

(C) COPYRIGHT IBM CORP. 1980, 2007.

Adding a Remote Journal

Adding a remote journal creates a remote journal on a target system or independent disk pool and associates that remote journal with the journal on the source system. This occurs if this is the first time the remote journal is being established for a journal. The journal on the source system can be either a local or remote journal.

If a remote journal environment has previously been established, adding a remote journal reassociates the remote journal on the target system with the journal on the source system.

You can establish and associate a remote journal on a target system with a journal on the source system by one of the following methods:

- System i Navigator.
- Add the Remote Journal (QjoAddRemoteJournal) API on the source system.
- Add the Remote Journal (ADDRMTJRN) command on the source system.

What Happens During Add Remote Journal Processing?

The processing that takes place as part of adding a remote journal includes the following:

- A check is performed on the target system to verify that the user profile adding the remote journal exists. A user profile with the same name as the user profile which is adding a remote journal must exist on the target system. If the profile does not exist on the target system, then an exception is signaled, and the processing ends.
- A check is performed to verify that the target system has a library by the same name as the library for the journal on the source system. If the library does not exist on the target system, then an exception is signaled, and the processing ends.
- A check is performed on the target system to determine if a journal by the same qualified name as the journal on the source system already exists. If a journal already exists, it can be used for the remainder of the add remote journal processing if it meets the following criteria:
 1. It is a remote journal.
 2. It was previously associated with this same source journal or part of the same remote journal network.
 3. The type of the remote journal matches the specified remote journal type.
- If a journal was found, but does not meet the preceding criteria, then an exception is signaled, and the processing ends. Otherwise, the remote journal is used for the rest of the add remote journal processing.
- If no journal is found on the specified target system, then a remote journal is created on the target system. The new remote journal has the same configuration, authority, and audit characteristics of the source journal. The journal that is created has a journal type of *REMOTE.

When adding the remote journal, you must specify the type of remote journal to add. The remote journal type influences the library redirection rules and other operational characteristics for the journal.

Guidelines For Adding a Remote Journal

You should observe the following guidelines for adding a remote journal:

- You can only associate a remote journal with a single source journal.
Note: The same remote journal can then have additional remote journals that are associated with it that are located on other target systems. This is the cascade configuration that is shown in Network configurations for remote journals.
- The remote journal will only have its attached receiver populated with journal entries that are replicated from the corresponding journal receiver on the source system. No journal entries can be directly deposited to a remote journal.
- A maximum of 255 remote journals can be associated with a single journal on a source system. This can be any combination of asynchronously maintained or synchronously maintained remote journals.

To Add a Remote Journal

The following is an example using the physical file QGPL/TESTPF setup to have remote journaling enabled to a second system.

1. Create the physical file.:

```
> CRTPF FILE(QGPL/TESTPF) RCDLEN(10)
File TESTPF created in library QGPL.
Member TESTPF added to file TESTPF in QGPL.
```
2. Create the local journal receiver and journals, and enable the journaling of the physical file created:

```
> crtjrncv jrncv(qgpl/jrcvrmt)
Journal receiver JRCVRMT created in library QGPL

> crtjrn jrn(qgpl/jrnrmnt) jrncv(qgpl/jrcvrmt) fixlendta(*job *usr *pgm *sysseq)
Journal JNRMT created in library QGPL

strjrnpf file(qgpl/testpf) jrn(qgpl/testpf)
1 of 1 files have started journaling
```
3. Add the remote journal:

```
> addrmtjrn rdb(sys2) srcjrn(qgpl/JNRMT) rmtjrntype(*TYPE2)
Remote journal JNRMT in QGPL was added
```
4. Activate the remote journaling:

```
> chgrmtjrn rdb(sys2) srcjrn(qgpl/jrnrmnt) jrnstate(*active)
Remote journal JNRMT in library QGPL was activated
```

Prepare Tables for Processing

You must perform the following tasks to prepare your tables for use in an Oracle GoldenGate environment for Db2 for i.

Ensuring Row Uniqueness for Tables

Oracle GoldenGate requires some form of 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:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. 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 those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration.

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.

4. If a table does not have an appropriate key, or if you prefer that the existing key(s) are not 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. See TABLE | MAP in *Parameters and Functions Reference for Oracle GoldenGate*.

Using KEYCOLS to Specify a Custom Key

If a table does not have one of the preceding types of row identifiers, or if you prefer those identifiers 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.

Preventing Key Changes

If you must add columns to the key that Extract is using as the row identifier for a table (primary key, unique key, KEYCOLS key, or all-column key) after Oracle GoldenGate has started processing journal data, follow these steps to make the change.

1. Issue the following command until it returns EOF, indicating that it has processed all of the existing journal data.

```
INFO EXTRACT group
```

2. Stop Extract.

```
STOP EXTRACT group
```

3. Make the change to the key.

4. Start Extract.

```
START EXTRACT group
```

Disabling Constraints on the Target

Triggers and cascade constraints must be disabled on the target tables or configured to ignore changes made by Replicat. Constraints must be disabled because Oracle GoldenGate replicates DML that results from a trigger or a cascade constraint. If the same trigger or constraint gets activated on the target table, it becomes redundant because of the replicated version, and the database returns an error. Consider the following example, where the source tables are emp_src and salary_src and the target tables are emp_targ and salary_targ.

1. A delete is issued for emp_src.
2. It cascades a delete to salary_src.
3. Oracle GoldenGate sends both deletes to the target.
4. The parent delete arrives first and is applied to emp_targ.
5. The parent delete cascades a delete to salary_targ.
6. The cascaded delete from salary_src is applied to salary_targ.
7. The row cannot be located because it was already deleted in step 5.

Maintaining Materialized Query Tables

To maintain parity between source and target materialized query tables (MQT), you replicate the base tables, but not the MQTs. The target database maintains the MQTs based on the changes that Replicat applies to the base tables.

The following are the rules for configuring these tables:

- Include the base tables in your TABLE and MAP statements.
- Do not include MQTs in the TABLE and MAP statements.
- Wildcards can be used in TABLE and MAP statements, even though they might resolve MQT names along with regular table names. Oracle GoldenGate automatically excludes MQTs from wildcarded table lists. However, any MQT that is explicitly listed in an Extract TABLE statement by name will cause Extract to abend.

Db2 for i: Supported Data Types, Objects, and Operations

This section contains information on supported data types, objects, and operations for Oracle GoldenGate on TimesTen.

Oracle GoldenGate on Db2 for i supports the filtering, mapping, and transformation of data unless otherwise noted in this documentation.

Oracle GoldenGate for Db2 for i runs remotely from a Linux system on a Db2 for i source system to capture data from the transaction journals for replication to a target system. To apply data to a target Db2 for i database, Oracle GoldenGate can run remotely from a Linux system. Oracle GoldenGate communicates with the IBM i system by means of an ODBC connection, and no Oracle GoldenGate software is installed on the Db2 for i target.

Note

The Db2 for i platform uses one or more **journals** to keep a record of transaction change data. For consistency of terminology in the supporting administrative and reference Oracle GoldenGate documentation, the terms "log" or "transaction log" may be used interchangeably with the term "journal" where the use of the term "journal" is not explicitly required.

Supported Db2 for i Data Types

Oracle GoldenGate supports all Db2 for i data types, except those listed in [Non-Supported Db2 for i Data Types](#).

Limitations of support

Extract fully supports the capture and apply of `TIMESTAMP(0)` through `TIMESTAMP(12)` when the output trail format is 19.1 or higher. Otherwise Extract truncates data from `TIMESTAMP(10)` through `TIMESTAMP(12)` to nanoseconds (maximum of nine digits of fractional time) and issues a warning to the report file.

Oracle GoldenGate supports timestamp data from 0001/01/03:00:00:00.000000 to 9999/12/31:23:59:59.999999. If a timestamp is converted from GMT to local time, these limits also apply to the resulting timestamp. Depending on the time zone, conversion may add or subtract hours, which can cause the timestamp to exceed the lower or upper supported limit.

The `DECFLOAT` data type supports the NaN, sNaN, Infinity/-Infinity special numeric values.

For more information on data types by databases that support special values, refer to [Special Values Supported by Database Data Types](#).

Non-Supported Db2 for i Data Types

Oracle GoldenGate does not support the following Db2 for i data types:

- XML
- DATALINK
- User-defined types

Supported Objects and Operations for Db2 for i

Oracle GoldenGate supports the following Db2 for i objects and operations.

- Extraction and replication of DML operations .
- Tables with the maximum row length supported by the database.
- Tables that contain up to the maximum number of columns that is supported by the database, up to the maximum supported column size.
- TRUNCATE operations are supported and are represented by `DELETE FROM` with no WHERE clause SQL statements and Clear Physical File Member (CLRPFM).
- Base tables underlying Materialized Query Tables, but not the MQTs themselves. The target database automatically maintains the content of the MQT based on the changes that are applied to the base tables by Replicat.
- Both Library (Native) names including members, and SQL names are allowed.

- Partitioned tables
- Supported options with `SEND EXTRACT SHOWTRANS || SKIPTRANS || FORCETRANS` command are:

Supported options with `SHOWTRANS`:

```
SHOWTRANS [transaction_ID] [COUNT n]
[DURATION duration unit] [TABULAR]
[FILE file_name] |
```

- Options for `SKIPTRANS` and `FORCETRANS`:

```
SKIPTRANS transaction_ID [FORCE]
FORCETRANS transaction_ID [FORCE]
```

- Limitations on Automatic Heartbeat Table support are as follows:
 - The `ADD HEARTBEATTABLE` command creates a new file called `ogghbfreq` in the Oracle GoldenGate installation directory. Do not delete this file because the `pas` heartbeat program reads the frequency values from it.
 - There is an extra executable in the Oracle GoldenGate build folder named `ogghb`.
 - An extra process named `ogghb` starts running on the IBM i system when the `ADD HEARTBEATTABLE` command runs until you disable the heartbeat with the `DELETE HEARTBEATTABLE` command. This process automatically restarts even if it is killed. To remove this process from the system, use the `DELETE HEARTBEATTABLE` command.
 - When using the `ALTER HEARTBEATTABLE` command to change the heartbeat frequency with the `PURGE_FREQUENCY` or `RETENTION_TIME` options, it might take some time for the new setting to take affect.
 - There is an initial delay of 30 seconds between `ADD HEARTBEATTABLE` and the first record is updated in the heartbeat seed table.
 - `[THREAD n]` and `[DETAIL]` is not supported.

Non-Supported Objects and Operations for DB2 for i

Oracle GoldenGate does not support the following objects or operations for DB2 for i.

- DDL operations
- Schema, table or column names that have trailing spaces
- Multiple instances of a database

Oracle GoldenGate Parameters Not Supported for DB2 for i

This section lists some of the Oracle GoldenGate configuration parameters that are not supported for the DB2 for i platform. For full descriptions of Oracle GoldenGate parameters and the databases they support, see *Oracle GoldenGate Parameters*.

```
BATCHSQL
BR
ASCIITOEBCDIC and EBCDICTOASCII
BINARYCHARS
LOBMEMORY
TRAILCHARSETEBCDIC
```

Supported Object Naming Conventions

Oracle GoldenGate supports SQL naming conventions and also supports native file system names in the format of *library/file(member)*.

For native (system) names, Oracle GoldenGate supports the normal DB2 for i naming rules for wildcarding, which allows *ALL or a partial name with a trailing asterisk (*) wildcard. For example:

- *library/*all(*all)*
- *library/a*(a*)*
- *library/abcde**

The member name is optional and may be left off. In that case, data for all of the members will be extracted, but only the library and file names will be captured and included in the records that are written to the trail. The result is that the data will appear to have come from only one member on the source, and you should be aware that this could cause integrity conflicts on the target if there are duplicate keys across members. To include the member name in the trail records, include the member explicitly or through a wildcarded member specification.

For SQL names, only the first member in the underlying native file is extracted in accordance with the normal operation of SQL on an Db2 for i system. For SQL names, Oracle GoldenGate supports the wildcarding of both table names and schema names. For instructions on wildcarding SQL names, see [Specifying Object Names in Oracle GoldenGate Input](#).

System Schemas for DB2 for i

The following schemas or objects are not automatically replicated by Oracle GoldenGate unless they are explicitly specified without a wildcard..

- "Q*"
- "SYSIBM"
- "SYSIBMADM"
- "SYSPROC"
- "SYSTOOLS"
- "#LIBRARY"
- "#RPGLIB"

Supported Character Sets

The default behavior of a DB2 for i Extract is to convert all character data to Unicode. The overhead of the performance of the conversion to UTF-8 for the text data has been substantially reduced. However, if you want to send data in its native character set you can use the parameter `DBOPTIONS USEDATABASEENCODING` to override the default behavior.

Db2 z/OS

With Oracle GoldenGate for Db2 z/OS, you can perform initial loads and capture transactional data from supported Db2 z/OS versions and replicate the data to a Db2 z/OS database or other supported Oracle GoldenGate targets, such as an Oracle Database.

Oracle GoldenGate for Db2 z/OS is installed and runs remotely on Linux, zLinux, or AIX.

Oracle GoldenGate for DB2 z/OS supports data filtering, mapping, and transformations unless noted otherwise in this documentation.

Prepare Database Users and Privileges for Db2 z/OS

Learn about creating database users and assigning privileges for Oracle GoldenGate for Db2 z/OS.

Oracle GoldenGate requires a database user account. Create this account and assign privileges according to the following guidelines.

The following table shows the required system privileges for a dedicated Oracle GoldenGate user created in Db2 z/OS:

Assign the Db2 privileges listed in the following table to the Extract and Replicat dedicated database users. These are in addition to any permissions that Db2 ODBC requires. Except where noted, all Extract privileges apply to initial-load and log-based Extract processes, .

The authorities listed in the following table can be provided by granting either SYSCTRL or DBADM plus SQLADM authority to the user running the Oracle GoldenGate processes.

Table 7-1 Privileges Needed by Oracle GoldenGate for Db2 z/OS

User privilege	Extract	Stored Procedure	Replicat
CONNECT to the remote Db2 subsystem.	Yes	Yes	Yes
MONITOR2 (This does not apply to initial-load Extract)	Y	-	N
SELECT ON the following SYSIBM tables: SYSTABLES SYSCOLUMNS SYSTABLEPART SYSKEYS SYSINDEXES SYSCOLAUTH SYSDATABASE SYSFORIGNKEYS SYSPARMS SYSRELS SYSROUTINES SYSSYNONYMS SYSTABAUTH SYSAUXRELS	Y	-	Y
SELECT on source tables ¹	Y	-	N
INSERT, UPDATE, DELETE on target tables	N	-	Y
CREATE TABLE ²	N	-	Y
EXECUTE on ODBC plan (default is DSNACLI)	Y	-	N
Privileges required by SQLEXEC procedures or queries that you will be using. ³	Y	-	N

- ¹ SELECT on source tables required only if tables contain LOB columns, or for an initial-load Extract, if used.
- ² Required if using ADD CHECKPOINTTABLE from the command line interface to use the database checkpoint feature.
- ³ SQLEXEC enables stored procedures and queries to be executed by an Oracle GoldenGate process.

Prepare Database Connection

Learn about configuring database connection, system and parameters settings for Oracle GoldenGate for Db2 z/OS.

Specifying the Number of Connection Threads

Every Oracle GoldenGate process makes a database connection. Depending on the number of processes that you will be using and the number of other Db2 connections that you expect, you might need to adjust the following Db2 system parameters on the DSNТИPE Db2 Thread Management Panel:

- MAX USERS (macro DSN6SYSP CTHREAD)
- MAX TSO CONNECT (macro DSN6SYSP IDFORE)
- MAX BATCH CONNECT (macro DSN6SYSP IDBACK)

Log reads use RRSAF, allow:

- Two Db2 threads per process for each of the following:
 - Extract
 - Replicat
 - The Admin Client command DBLOGIN (logs into the database)
 - DEFGEN utility (generates data definitions for column mapping)
- One extra Db2 thread for Extract for IFI calls.
- One extra Db2 thread for each SQLEXEC parameter statement that will be issued by each Extract and Replicat process.

Ensuring ODBC Connection Compatibility

To ensure that you configure the Db2 ODBC initialization file correctly, follow the guidelines in the *Db2 UDB z/OS ODBC Guide and Reference* manual. One important consideration is the coding of the open and close square brackets (the [character and the] character). The square bracket characters are "variant" characters that are encoded differently in different coded character set identifiers (CCSID), but must be of the IBM-1047 CCSID in the ODBC initialization file. Db2 ODBC does not recognize brackets of any other CCSID. Note the following:

- The first (or open) bracket must use the hexadecimal characters X'AD' (0xAD).
- The second (or close) bracket must use the hexadecimal characters X'BD' (0xBD).

To set the correct code for square brackets, use any of the following methods.

- Use the hex command in OEDIT and change the hex code for each character appropriately.
- Use the iconv utility to convert the ODBC initialization file. For example, to convert from CCSID IBM-037 to IBM-1047, use the following command:

```
iconv -f IBM-037 -t IBM-1047 ODBC.ini > ODBC-1047.ini
```

```
mv ODBC-1047.ini ODBC.ini
```

- Change your terminal emulator or terminal configuration to use CCSID IBM-1047 when you create or alter the file.

Database Configuration

Learn about database configuration settings for Oracle GoldenGate for Db2 z/OS. The database settings are required for Oracle GoldenGate.

Install Extract Components on Db2 z/OS

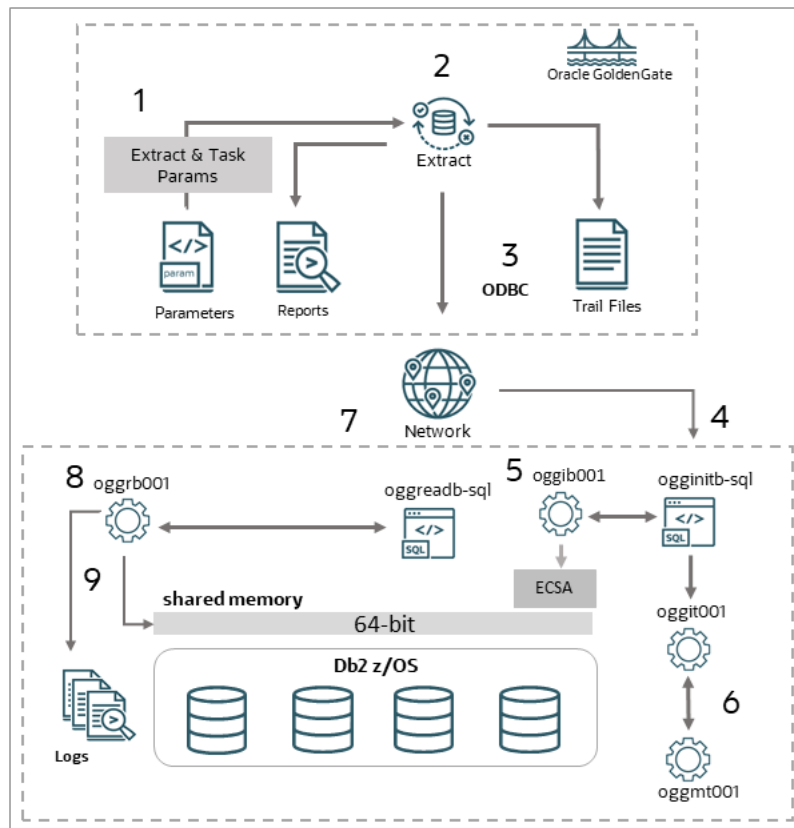
The Oracle GoldenGate Db2 z/OS Extract uses SQL objects to access and read the Db2 log. These Oracle GoldenGate Db2 z/OS objects require a minimum hardware platform of zEC12, a minimum operating system release of 2.4, and a minimum Db2 release of 12.1. The components consist of executable load modules, SQL stored procedures and functions, and external programs called via the stored procedures. These components are:

1. External authorized programs include the following:
 - a. oggib001 – Initialization and utility program
 - b. oggrb001 – Log read program functionality
 - c. oggmt001 – Stand-alone program that monitors ECSA and 64-bit memory
 - d. oggjt001 – Setup program for the oggmt001 startup JCL run from oggib001 program
 - e. oggfr001 – Utility for use by a DBA under guidance from Oracle Support
2. SQL stored procedure and function templates are included in the SQL script `zOS_OGG_Setup_Template.sql` with the `OGGINITB` and `OGGREADB` SQL.
3. JCL procedure, `oggtask.jcl`

Note

The external names for the SQL and JCL name values are the default, which you can edit and update. This process is discussed in the subsequent sections.

The Replication Process for Db2 z/OS Extract figure illustrates the replication process for the Db2 z/OS Extract and its mainframe components.

Figure 7-1 Replication Process for Db2 z/OS Extract

The process starts and runs as shown in the figure above, using the numbers 1 through 9. These steps are listed below:

1. Extract reads the parameters, including the JCL parameters, from the parameter file created during installation.
2. Extract reports the startup information and prepares to write the trail files.
3. ODBC is used to gather information from the Db2 database and start replication.
4. The OGGINITB SQL stored procedure starts to prepare the shared memory and to gather other data needed for replication.
5. The OGGIB001 external program called by the SQL stored procedure starts the memory monitor task using the OGGJT001 job setup program.
6. The OGGMT001 memory monitor task starts monitoring the ECSA and 64-bit shared memory.
7. The OGGREADB SQL Function calls the external program OGGRB001.
8. The OGGRB001 external program repeatedly calls the Db2 log read program to create a result set that returns 1 to many log record buffers to the Extract.
9. When a log record result set is complete, OGGRB001 ends after sending the result set to the Extract.

Extract repeats steps 7 to 9 until shut down or abnormal termination. If the memory task fails to start, OGGIB001 program returns a flag indicating there was a JCL error or setup issue and Extract begins to manage its memory. If the memory task starts properly, the memory task tests constantly changing fields in the 48-byte ECSA shared memory. These fields stop

changing if the Extract terminates for any reason. At that point, the memory manager waits in case the Extract or network is slow and releases the memory before shutting down after a configured time limit.

To install the components needed for Oracle GoldenGate for Db2 z/OS Extract:

1. Ensure that a library (PDSE) exists on the Db2 z/OS system and ensure that an entry for it is made in the authorized library list. This library is the location where the Oracle GoldenGate external program objects will reside.
2. Ensure that an APF-authorized WLM procedure references the PDSE from the preceding step. Oracle recommends that NUMTCB value for the WLM environment be 10-40 for stored procedures. The NUMTCB value depends on the maximum number of Extracts that are running concurrently against the database and on how much throughput each Extract requires. If you want flexibility in setting NUMTCB, specify it in the startup JCL for the WLM, but not in the creation panel.
3. You can set up security for the WLM application environments and for creating stored procedures by completing the following:
 - a. (Optional) Specify which WLM-established address spaces can run stored procedures. If you do not complete this step, then any WLM-established address space can run stored procedures.
 - b. Grant users access to create procedures in specific WLM address spaces.
 - c. Grant user access to create procedures in specific schemas. Use the GRANT statement with the CREATIN option for the appropriate schema.
 - d. Grant user access to create packages for procedures in specific collections. Use the GRANT statement with the CREATE option for the appropriate collection.
 - e. Grant access to refresh the WLM environments to the appropriate people.
 - f. Add additional RACF authority to the appropriate people, allowing the WLM procedures to start the memory manager job.
4. Ensure the ID used to run the WLM startup JCL procedure has permission to use RRSAPF. Each time one of the Db2 WLM address spaces starts, it uses RRSAPF to attach to Db2. See the [Db2 12 for z/OS Installation and Migration Guide](#).
5. In the Linux or UNIX installation of Oracle GoldenGate for Db2 z/OS, there is a ZIP file located in the Oracle GoldenGate installation directory called z0Sutils.zip, which contains a file called z0SPrograms.zip. Unzip z0SPrograms.zip to z0SPrograms.tar and copy z0SPrograms.tar in binary mode to your Db2 z/OS system into an HFS directory.
6. On your Db2 z/OS system in USS or OMVS, change directories to the directory containing z0SPrograms.tar.
7. Restore the objects with the command: `tar -xovf z0SPrograms.tar`.
8. Copy the objects to the authorized PDSE. Use the `cp -X ogg[irmj][abt][0-9]* "///'authorized_PDSE_name'"` where `authorized_PDSE_name` is the name of the APF authorized PDSE, intended for the Oracle GoldenGate objects. Using this command installs the objects with the default names.

Note

In this command, the copy target is double-quote forward-slash single-quote authorized PDSE name single-quote double quote. The -X is an uppercase capital X *not* a lowercase x.

9. Installing the scripts with different names allows you to conform with system protocols, or it allows you to run multiple versions of Oracle GoldenGate. To install the scripts with different names, creating a shell script that renames the programs before copying them to the PDSE is recommended. An example of the shell script is given in the following code snippet.

```
#!/bin/bash
# Copy new programs renaming them to version 21.12 names.
cp oggib001 oggi2112
cp oggrb001 oggr2112
cp oggmt001 oggm2112
cp oggjt001 oggj2112
cp -X oggi2112 '///SYS4.WLMDSNA.AUTHLOAD'
cp -X oggr2112 '///SYS4.WLMDSNA.AUTHLOAD'
cp -X oggm2112 '///SYS4.WLMDSNA.AUTHLOAD'
cp -X oggj2112 '///SYS4.WLMDSNA.AUTHLOAD'
```

You can run the script using `chmod +x scriptname` command. You can copy and reuse this script for new versions.

10. You must create the SQL procedures using your SQL tool of choice so that Oracle GoldenGate can call the Extract objects. The Oracle GoldenGate stored procedures should have permission granted to only those users that use them for replication.

An example SQL script template in the Oracle GoldenGate install directory contains the SQL statements to set up the stored procedure and function on the Db2 z/OS instance. The SQL script `zOS_OGG_Setup_Template.sql` contained in `zOSutils.zip` is for Db2 v12.1 and higher and can run from any SQL tool on any platform that can connect to your Db2 z/OS instance. This script must run on the Db2 instance that you use with your Extract. The script provided in the remote installation directory is in ASCII format.

Edit the following line before running the scripts:

- Modify the `WLM ENVIRONMENT` line to use the correct name for the WLM environment that you will use for Oracle GoldenGate.

Note

The oggifi0001 schema name is configurable using the TRANLOGOPTIONS REMOTESCHEMA *schemaname* Extract parameter. The procedure and function names, OGGINITB and OGGREADB, in the template are not configurable. You can rename each external name in the scripts and the PDSE if the script names and the PDSE object names match. Changing these names is part of the procedure that allows migration to new versions or if specific naming procedures must be adhered to on Db2 z/OS.

The following table contains a check list of components that you may wish to edit and/or update:

Table 7-2 List of Editable Components

Component	From	Rename	Where
oggib001	tar file		authorized PDSE
oggrb001	tar file		authorized PDSE
oggmt001	tar file		authorized PDSE & proc library
oggjt001	tar file		authorized PDSE & Extract parm
oggpr001	tar file		procedure library & Extract parm
proclib	MVS		add Extract parm if needed
step libraries	MVS		WLM and oggpr001 procedure library
remoteschema			zOS_OGG_Setup_Template.sql and Extract parm
WLM name	MVS		zOS_OGG_Setup_Template.sql
external program			zOS_OGG_Setup_Template.sql

Note

Remember to perform all these steps after every new patch installation.

Use Shared Memory Manager for Extract

Oracle GoldenGate Extract starts a separate task, or job, from the WLM to monitor shared memory usage. This job is a minimal task that runs in MVS, but it is **not** a WLM. The username of this script starts under must have permissions to execute when starting from the WLM (usually using something like RACF). This monitored shared memory consists of a small 48 to 64 byte ECSA area, and a larger 64-bit area based on the Extract buffer size.

Specific fields in shared memory get updated for every read performed by the Extract. These fields are updated whether or not the script returns any data. The monitor checks those fields to ensure the Extract has not become inactive. If the Extract is inactive, the shared memory is

released, and the monitor ends. You can control the Memory Manager using the `remote_memory_options` parameter in the Extract's parameter file.

You can specify multiple sub-parameters to configure the monitor task. You can configure the wait interval and inactive time the monitor uses by specifying sub-parameters of the remote memory options, as shown in the following example:

```
remote_memory_options wait_interval 2000 inactive_time 01:00
```

The wait interval is expressed in hundredths of seconds in the example and causes the monitor to wait 20 seconds between each memory check. If the monitor has checked for 1 hour (format HH:MM) and the Extract is still inactive, then the monitor will shut down after releasing the shared memory. If the Extract returns to an active state during that hour, the monitor will reset its state and continue monitoring.

The `wait_interval` can have values from 100 to 6000 and the default is 1000. The `inactive_time` can be from 00:10 to 12:00 and the default is 00:30. If the monitor does not start properly, the Extract displays a warning message in the Extract report and the Extract continues the processing. The Extract will attempt to release ECSA memory when it shuts down.

The remote memory parameter has three options to make this feature work. The syntax for these parameters is:

- `task_procedure proc name`
- `task_library proc library`
- `task_setup task setup program`

Example:

```
remote_memory_options task_procedure OGGPR001  
remote_memory_options task_library TEST.PROCLIB  
remote_memory_options task_setup OGGJT001
```

You may specify multiple options in a single command, as shown below:

```
remote_memory_options task_procedure OGGPR001 task_library TEST.PROCLIB  
task_setup OGGJT001
```

 Note

The values for the remote memory parameter are case insensitive.

The default values are procedure name OGGPR001 and the task setup program OGGJT001. There is no default for task library as the procedure might be installed in one of the MVS system default procedure libraries. The task library parameter is only needed if the procedure is not in a system default library.

The memory task will start with a simple JOB card and an EXEC procedure name with parameters passed from the Extract. Some z/OS systems may require various other

parameters on the job card. The JOB parameters can also be modified using the remote memory parameter, as shown in the example given below.

```
remote_memory_options task_jobname [valid MVS job name (see below)]
remote_memory_options task_acct_info [valid MVS acct value (see below)]
remote_memory_options task_programmer [valid MVS programmer name, Can use
single quotes]
remote_memory_options task_class [valid MVS job class A to Z or 0 to 9]
remote_memory_options task_msgclass [valid MVS msgclass A to Z or 0 to 9]
remote_memory_options task_msglevel [valid MVS message level n or (,n) or
(n,n) n=valid digit]
remote_memory_options task_priority [valid MVS priority 0-15]
```

You can specify the JOB name using two valid characters and an asterisk, such as AA*. The default JOB name is GG*. The asterisk is replaced by six random numbers when it is specified. Otherwise, if you specify a one to eight byte character name, it must be a valid MVS job name.

Specify account values in any of the following valid MVS formats:

- OTXI
- 'MY ACCT'
- (ACCT,1234,ABC)

For parameters, like acct_info and programmer, that allow special characters, enclose those in single quotes. In addition, the MVS rules about using double single quotes or ampersands within quotes continue to apply. The Extract does minimal validation for these parameters and leaves the complete validation to the MVS process. Extract will accept the first one if you specify duplicate parameters and ignore any duplicates.

A sample procedure JCL file will be included in the z0SPrograms.zip file. This JCL does not replace the WLM procedure. The monitor, created in the WLM, uses the JCL to run in MVS. A JCL procedure allows more flexibility if you add commands or JCL that support job output archiving or other procedures.

Note

During installation and setup for the memory manager, temporarily turning off the memory manager may be necessary when you must wait for RACF setup issues, initiator usage, job class usage, or other similar issues related to the memory manager.

The Extract will operate in its legacy mode in this instance and release ECSA, and the 64-bit shared memory. The Extract shows the release of shared memory at the end of the Extract report.

Oracle does not recommend this as a permanent solution. When used, the user is responsible for monitoring the release of ECSA and 64-bit memory. The parameter to turn on this feature begins with an underscore to remind the user that the system is running in this mode. Other remote memory options can be left in place and used by removing the option to turn memory

management off. The remote memory management parameter defaults to being on, and it is specified as shown:

```
remote_memory_options _remote_memory_on
remote_memory_options _remote_memory_off
```

The JCL has the following format:

```
//*=====
/* EXAMPLE JCL FOR RUNNING THE COMMON MEMORY MONITOR PROCEDURE
/* ADDRESS SPACE NEEDING AN AUTHORIZED LOAD LIBRARY
/* NOTE: THE PROGRAM OGGMT001 CAN BE RENAMED IN THE LIBRARY BUT THE
/*      NEW NAME MUST MATCH THE PROGRAM NAME IN THIS JCL
/*=====
//OGGDSNNA PROC RGN=0K TR=,EX=,MEM=,LEN=,SEC=,DUR=,VER=
//OGGDSNNX EXEC PGM=OGGMT001,REGION=&RGN,TIME=NOLIMIT,
// PARM='&TR &EX &MEM &LEN &SEC &DUR &VER'
/*-----
/* REPLACE &PREFIX.**.AUTHLOAD LIBRARIES WITH SITE SPECIFIC FILE(S)
/* ALSO REPLACE THE CEE LIBRARY WITH SITE SPECIFIC FILE
/* DSNN COULD REPRESENT A DB2 SPECIFIC LOAD LIBRARY IF ONE EXISTS
/*-----
//STEPLIB DD DISP=SHR,DSN=&PREFIX..WLMDSNN.USER.AUTHLOAD
//        DD DISP=SHR,DSN=CEE.SCEERUN
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
```

Modify the libraries marked with PREFIX so that they work in your system. If you renamed the program OGGMT001 you copied from the zOSPrograms.tar file, you must change it in the JCL. The null parameters on the PROC statement are there for information purposes. The job OGGJT001 setup program supplies those values using information passed from the Extract. You may also specify as many step library dataset names as required. The JCL procedure supplied in the zOSPrograms.tar file gives an example using more than one step library.

Support Globalization Functions

Oracle GoldenGate provides globalization support, which you should consider when using this support.

Replicating From a Source that Contains Both ASCII and EBCDIC

When replicating to or from a Db2 source system to a target with a different character set, some consideration should be given to the encoding of the character data on the Db2 source if it contains a mix of ASCII and EBCDIC data. Character set conversion by any given Replicat requires source data to be in a single character set.

The source character set gets specified in the trail header. Thus, the Oracle GoldenGate trail can contain either ASCII or EBCDIC data, but not both. Unicode tables are processed without particular configuration and are exempt from the one-character set requirement.

For a source that contains both character encoding types, you have the following options:

- You can use one Extract for all of your tables and have it write the character data to the trail as either ASCII or as EBCDIC.

- You can use different Extracts: one Extract to write the ASCII character data to a trail, and another Extract to write the EBCDIC character data to a different trail. You then associate each trail with its own Extract and Replicat process, so that the two data streams are processed separately.

To output the correct character set in either of those scenarios, use the `TRAILCHARSETASCII` and `TRAILCHARSETEBCDIC` parameters. The default is `TRAILCHARSETEBCDIC`. Without these parameters, ASCII and EBCDIC data are written to the trail as-is. When using these parameters, note the following:

- If used on a single-byte Db2 subsystem, these parameters cause Extract to convert all of the character data to either the ASCII or EBCDIC single-byte CCSID of the subsystem to which Extract is connected, depending on which parameter is used (except for Unicode, which is processed as-is).
- If used on a multi-byte Db2 subsystem, these parameters cause Extract to capture only ASCII or EBCDIC tables (and Unicode). Character data gets written in either the ASCII or EBCDIC mixed CCSID (depending on the parameter used) of the Db2 z/OS subsystem to which Extract is connected.

Specifying Multi-Byte Characters in Object Names

If the name of a schema, table, column, or stored procedure in a parameter file contains a multi-byte character, use double quotes on the name.

For more information about specifying object names, see [Specifying Object Names in Oracle GoldenGate Input](#).

Prepare Tables for Processing

You must perform the following tasks to prepare your tables for use in an Oracle GoldenGate environment for Db2 z/OS.

Disable Triggers and Cascade Constraints

Disable triggers, cascade delete constraints, and cascade update constraints on the target tables, or alter them to ignore changes made by the Oracle GoldenGate database user. Oracle GoldenGate replicates DML that results from a trigger or cascade constraint. If the same trigger or constraint gets activated on the target table, it becomes redundant because of the replicated version, and the database returns an error. Consider the following example, where the source tables are `emp_src` and `salary_src` and the target tables are `emp_targ` and `salary_targ`.

- A delete gets issued for `emp_src`.
- It cascades a delete to `salary_src`.
- Oracle GoldenGate sends both deletes to the target.
- The parent delete arrives first and is applied to `emp_targ`.
- The parent delete cascades a delete to `salary_targ`.
- The cascaded delete from `salary_src` is applied to `salary_targ`.
- The row gets deleted in step 5 and cannot be located.

Ensure Row Uniqueness for 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 exists in the TABLE or MAP statement, Oracle GoldenGate selects a row identifier to use in the following order of priority:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. 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 those that Oracle GoldenGate does not supported in a key or those that are excluded from the Oracle GoldenGate configuration.

Note

If there are other non-usable keys on a table or no keys 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 more extensive, less efficient WHERE clause.

4. If a table does not have an appropriate key, or if you prefer that the existing key(s) are not 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 and the Replicat MAP parameters. The specified key will override any existing primary or unique key that Oracle GoldenGate finds. See [TABLE | MAP](#) in *Reference for Oracle GoldenGate*.

Using KEYCOLS to Specify a Custom Key

If a table does not have one of the preceding types of row identifiers, or if you prefer those identifiers 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 and the Replicat MAP parameters. The specified key will override any existing primary or unique key that Oracle GoldenGate finds. For more information, see [Reference for Oracle GoldenGate](#).

Handle Tables with ROWID Columns

Any attempt to insert into a target table that includes a column with a data type of ROWID GENERATED ALWAYS (the default) will fail with the following ODBC error:

```
ODBC error: SQLSTATE 428C9 native database error -798. {DB2 FOR OS/390}{ODBC DRIVER}{DSN08015} DSNT408I SQLCODE = -798, ERROR: YOU CANNOT INSERT A VALUE INTO A COLUMN THAT IS DEFINED WITH THE OPTION GENERATED ALWAYS. COLUMN NAME ROWIDCOL.
```

You can do one of the following to prepare tables with ROWID columns to be processed by Oracle GoldenGate:

- Ensure that any ROWID columns in target tables are defined as GENERATED BY DEFAULT.

- If it is not possible to change the table definition, you can work around it with the following procedure.

To workaround ROWID GENERATE ALWAYS:

1. For the source table, create an Extract TABLE statement, and use a COLSEXCEPT clause in that statement that excludes the ROWID column. For example:

```
TABLE tab1, COLSEXCEPT (rowidcol);
```

The COLSEXCEPT clause excludes the ROWID column from being captured and replicated to the target table.

2. For the target table, ensure that Replicat does not attempt to use the ROWID column as the key. This can be done in one of the following ways:

- Specify a primary key in the target table definition.
- If a key cannot be created, create a Replicat MAP parameter for the table, and use a KEYCOLS clause in that statement that contains any unique columns except for the ROWID column. Replicat will use those columns as a key. For example:

```
MAP tab1, TARGET tab1, KEYCOLS (num, ckey);
```

Maintaining Materialized Query Tables

To maintain parity between source and target materialized query tables (MQT), you replicate the base tables, but not the MQTs. The target database maintains the MQTs based on the changes that Replicat applies to the base tables.

The following are the rules for configuring these tables:

- Include the base tables in your TABLE and MAP statements.
- Do not include MQTs in the TABLE and MAP statements.
- Wildcards can be used in TABLE and MAP statements, even though they might resolve MQT names along with regular table names. Oracle GoldenGate automatically excludes MQTs from wildcarded table lists. However, any MQT that is explicitly listed in an Extract TABLE statement by name will cause Extract to abend.

Prepare Db2 z/OS Transaction Logs for Oracle GoldenGate

Learn to configure the Db2 transaction logging to support data capture by Oracle GoldenGate Extract.

Oracle GoldenGate can capture Db2 transaction data from the active and archived logs. Follow these guidelines to configure the logs so that Extract can capture data.

To enable change capture for Oracle GoldenGate for Db2 z/OS, see [Enable TRANDATA for Non-Oracle Databases](#)

Enable Access to Log Records

Activate Db2 Monitor Trace Class 1 ("TRACE(MONITOR) CLASS(1) ") so that Db2 allows Extract to read the active log. The default destination of OPX is sufficient, because Oracle GoldenGate does not use a destination.

To Start the Trace Manually

1. Log on to Db2 as a Db2 user with the TRACE privilege or at least SYSOPR authority.

2. Issue the following command:

```
start trace(monitor) class(1) scope(group)
```

To Start the Trace Automatically When Db2 is Started

Do either of the following:

- Set MONITOR TRACE to "YES" on the DSNTIPN installation tracing panel.
- Set 'DSN6SYSP MON=YES ' in the DSNTIJUZ installation job, as described in the *Db2 UDB Installation Guide*.

Note

The primary authorization ID, or one of the secondary authorization IDs, of the ODBC plan executor must also have the MONITOR2 privilege.

Size and Retain Logs

More data gets logged when tables are defined with DATA CAPTURE CHANGES than when they are defined with DATA CAPTURE NONE. If any of the following is true, you might need to increase the number and size of the active and archived logs.

- Your applications generate large amounts of Db2 data.
- Your applications have infrequent commits.
- You expect to stop the Extract for long periods.
- Your network is unreliable or slow.

Use the DSN6LOGP MAXARCH system parameter in the DSNTIJUZ installation job to control log retention.

Retain enough log data so that Extract can start again from its checkpoints after you stop it or after an unplanned outage. Extract must have access to the log that contains the start of the oldest uncommitted unit of work and all logs after that.

If data that Extract needs during processing was not retained, either in online or archived logs, one of the following corrective actions might be required:

- Alter Extract to capture from a later point in time for which log data is available (and accept possible data loss on the target).
- Resynchronize the source and target tables, and then start the Oracle GoldenGate environment again.

Note

The IBM documentation makes recommendations to improve the performance of log reads. In particular, you can use large log output buffers, large active logs, and make archives to disk.

Use Archive Logs on Tape

Oracle GoldenGate can read Db2 archive logs on tape, which degrades performance. For example, Db2 reserves taped archives for a single recovery task, leading to Extract not being able to read an archive tape that is being used to recover a table until the recovery completes. You could use DFHSM or an equivalent tool to move the archive logs seamlessly between online DASD storage and tape, but Extract will have to wait until the transfer completes. Delays in Extract processing increase the latency between source and target data.

If the Oracle GoldenGate Extract can or will be reading from tape, either physical tape or virtual tape, it is highly recommended to ensure that the following ZPARMs are considered:

DEALLCT

The DEALLCT subsystem parameter determines the length of time that an archive read tape unit is to be allowed to remain unused before it is deallocated.

Update: option 43 on panel DSNTIPB

DSNZPxxx: DSN6LOGP DEALLCT

Oracle recommends that DEALLCT be set to 0, 1 (1 second deallocation delay). The default is to deallocate immediately. This will prevent Db2 from immediately reallocating a tape between reads from Extract.

MAXRTU

The MAXRTU subsystem parameter specifies the maximum number of dedicated tape units that can be allocated to concurrently read archive log tape volumes.

Update: option 43 on panel DSNTIPB

DSNZPxxx: DSN6LOGP MAXRTU

The default is 2 tape units, but it should be set to the number of expected processes that may read concurrently from tape+1. For example, if there are 4 Extracts and 2 other applications that may read from tapes, then MAXRTU should be at least 7 (4 + 2 + 1).

For best performance and least delays when reading from tape, Oracle GoldenGate Extract should use:

TRANLOGOPTIONS BUFSIZE 33554432

This will be an indication to Extract to read up to 32MB at a time from the transaction log, instead of the default of 1MB per read. This setting will not increase the memory allocations Oracle GoldenGate uses on z/OS, either ECSA (64 bytes), or 64-bit shared memory (1MB). Db2 transient memory allocations may increase while the Extract reads the large buffers.

Also, in the Extract parameter file, prior to the DBLOGIN parameter, the following may also be used to provide the ability to further tune the Db2 driver TCP communication buffering. For example:

```
SETENV(DB2SOSNDBUF='524288')  
SETENV(DB2SORCVBUF='524288')
```

This would inform the Db2 driver to use 512KB for the send and receiver buffers. Other values could be used, and would be tunable for each install.

Control Log Flushes

When reading the transaction log, Extract does not process a transaction until it captures the commit record. If the commit record is on a data block that is not full, it cannot be captured until

more log activity is generated to complete the block. The API that Extract uses to read the logs only retrieves full physical data blocks.

A delay in receiving blocks that contain commits can cause latency between the source and target data. If the applications are not generating enough log records to fill a block, Extract generates log records by issuing SAVEPOINT and COMMIT statements until the block fills up one way or the other and is released.

In a data sharing group, each API call causes Db2 to flush the data blocks of all active members, eliminating the need for Extract to perform flushes.

To prevent Extract from performing flushes, use the Extract parameter TRANLOGOPTIONS with the NOFLUSH option.

Db2 z/OS: Supported Data Types, Objects, and Operations

This section contains support information for Oracle GoldenGate for Db2 z/OS database.

Oracle GoldenGate for Db2 for z/OS supports capture and delivery of initial load and transactional data for supported Db2 for z/OS database versions.

Oracle GoldenGate for Db for z/OS supports the mapping, filtering, and transformation of source data, unless noted otherwise in this document, along with replicating data derived from other source databases supported by Oracle GoldenGate, into Db2 for z/OS databases.

Oracle GoldenGate for Db2 for z/OS is installed and runs remotely on Linux and zLinux.

Supported Db2 z/OS Data Types

Here is the list of the Db2 for z/OS data types that Oracle GoldenGate supports and any limitations of this support.

- Oracle GoldenGate does not perform character set conversion for columns that could contain multi-byte data. These include GRAPHIC, VARGRAPHIC and DBCLOB data types, as well as CHAR, VARCHAR, and CLOB for tables defined with ENCODING_SCHEME of 'M' (multiple CCSID set or multiple encoding schemes) or 'U' (Unicode). Such data is only supported if the source and target systems are the same CCSID.
- Oracle GoldenGate supports ASCII, EBCDIC, and Unicode data format. Oracle GoldenGate converts between ASCII and EBCDIC data automatically, however, it does not convert Unicode.
- Oracle GoldenGate supports most Db2 data types except those listed in [Non-Supported Db2 for z/OS Data Types](#).

Limitations of Support

- The support of range and precision for floating-point numbers depends on the host machine. The precision is generally accurate to 16 significant digits, but you should review the database documentation to determine the expected approximations. Oracle GoldenGate rounds or truncates the values that exceed the supported precision.
- Oracle GoldenGate does not support the filtering, column mapping, or manipulation of large objects greater than 4K in size. You can use the full Oracle GoldenGate functionality for 4K or smaller objects.
- Oracle GoldenGate supports the default TIMESTAMP and the TIMESTAMP with TIMEZONE to up to 9 digit fractional value, but no further.
- The DECFLOAT data type supports the NaN, sNaN, Infinity/-Infinity special numeric values.

For more information on data types by databases that support special values, refer to [Special Values Supported by Database Data Types](#).

Non-Supported Db2 for z/OS Data Types

Here is the list of Db2 for z/OS data types that Oracle GoldenGate does not support. Data that is not supported may affect the integrity of the target data with the source data.

- Negative dates
- User-defined types
- XML

Supported Objects and Operations for Db2 z/OS

Here is the list of database objects and types of operations that Oracle GoldenGate supports.

- The extraction and replication of DML operations on Db2 for z/OS tables supports the maximum row length of Db2 for z/OS and LOB columns, which can be up to 2GB in size.
- INSERT operations from the IBM LOAD utility are supported for change capture if the utility is run with LOG YES and SHRLEVEL CHANGE, and the source tables that are being loaded have DATA CAPTURE CHANGES enabled (required by Oracle GoldenGate). Also, specify or map these tables in the Oracle GoldenGate Extract configuration. Oracle GoldenGate also supports initial loads with the LOAD utility to instantiate target tables during initial synchronization.
- Oracle GoldenGate supports the maximum number of table columns supported by the database.
- Oracle GoldenGate supports the maximum column size supported by the database.
- Oracle GoldenGate supports extracting and replicating data stored using Db2 data compression (CREATE TABLESPACE COMPRESS YES).
- Capture from temporal history tables is supported.
- TRUNCATE TABLE is supported, but because this command issues row deletes to perform the truncate, they are shown in Oracle GoldenGate statistics as such, and not as a truncate operation. The Replicat process uses the DELETE operation without the WHERE clause to replicate the TRUNCATE statement.
- TRUNCATES are always captured from a Db2 for z/OS source, but can be ignored by Replicat if the IGNORETRUNCATES parameter is used in the Replicat parameter file.
- UNICODE columns in EBCDIC tables are supported.
- Supported options with SHOWTRANS

```
SHOWTRANS [transaction_ID] [COUNT n]
[DURATION duration unit]
[FILE file_name] |
```

Do not specify, transaction_ID and count or transaction_ID and duration together.

- Options supported with SKIPTRANS and FORCETRANS:

```
SKIPTRANS transaction_ID
[FORCE] FORCETRANS transaction_ID [FORCE]
```

Non-Supported Objects and Operations for Db2 z/OS

Oracle GoldenGate does not support the following objects and operations on Db2 z/OS:

- Extraction or replication of DDL operations
- Clone tables
- User-supplied Db2 exit routines that manipulate data remain unsupported, such as:
 - Compression
 - Date and time routines
 - Edit routines (CREATE TABLE EDITPROC) in Db2 release versions less than 13.1
 - Validation routines (CREATE TABLE VALIDPROC)
- Replicating with BATCHSQL is not fully functional for Db2 z/OS. Non-insert operations are not supported so any update or delete operations will cause Replicat to drop temporarily out of BATCHSQL mode. The transactions will stop and errors will occur.

MySQL

With Oracle GoldenGate for MySQL, you can perform initial loads and capture transactional data and table changes from supported MySQL versions and replicate the data and table changes to a MySQL database or replicate the data to other supported Oracle GoldenGate targets, such as an Oracle Database.

Oracle GoldenGate for MySQL supports data filtering, mapping, and transformations unless noted otherwise in this documentation.

This part describes tasks for configuring and running Oracle GoldenGate for MySQL and supported variants, such as MariaDB, Amazon RDS for MySQL, and Amazon Aurora MySQL.

Prepare Database Users and Privileges for Oracle GoldenGate for MySQL

Requirements for the database user for Oracle GoldenGate processes are as follows:

- Create a database user that is dedicated to Oracle GoldenGate. It can be the same user for all the Oracle GoldenGate processes that must connect to a database.
- 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, the Oracle GoldenGate database user.
- Keep a record of the database users credentials to set up database connection from Oracle GoldenGate. These credentials are required when setting up the USERIDALIAS for database login (DBLOGIN) from Oracle GoldenGate.
- The Oracle GoldenGate user requires read access to the INFORMATION_SCHEMA database.
- The Oracle GoldenGate user requires the following user privileges.

Privilege	Source Extract	Target Replicat	Purpose
SELECT	Yes	Yes	Connect to the database and select object definitions

Privilege	Source Extract	Target Replicat	Purpose
REPLICATION SLAVE	Yes	NA	Connect and receive updates from the replication master's binary log
CREATE CREATE VIEW EVENT INSERT UPDATE DELETE	Yes	Yes	Source and target database heartbeat and checkpoint table creation, and data record generation and purging
DROP	Yes	Yes	Dropping a Replicat checkpoint table or deleting a heartbeat table implementation
EXECUTE	Yes	Yes	To execute stored procedures
INSERT, UPDATE, DELETE on target tables	NA	Yes	Apply replicated DML to target objects
DDL privileges on target objects (if using DDL support)	NA	Yes	Issue replicated DDL on target objects

User Privileges Required for Local Capture

To capture binary log events by the local capture, an Administrator must provide the following privileges to the Extract user:

- Read and Execute permissions for the directory where the MySQL configuration file (`my.cnf`) is located.
- Read permission for the MySQL configuration file (`my.cnf`).
- Read and Execute permissions for the directory where the binary logs are located.
- Read and Execute permission for the `tmp` directory. The `tmp` directory is `/tmp`.

Prepare Database Connection

Learn about configuring database connections for Oracle GoldenGate for MySQL.

Oracle GoldenGate for MySQL is packaged with a MySQL client and uses that client to connect to MySQL databases. Connections are established by using a direct connection and supplying the database server host, port, database name, and other information.

Connections are created manually by adding a database connection to the Administration Service's web interface or through the Admin Client.

To set up the database connection from Oracle GoldenGate for a MySQL deployment, see [Add Database Connections](#).

Configure a Two-way SSL Connection for MySQL Extract and Replicat

To use Mutual TLS (mTLS or two-way SSL) with Oracle GoldenGate for MySQL Extract and Replicat, you need to supply the full paths of the certificate authority (`ca.pem`), the client

certificate (client-cert.pem) and the client key (client-key.pem) files to the capture and delivery.

To know more about generating the certificate files, see:

<https://dev.mysql.com/doc/refman/8.0/en/creating-ssl-rsa-files-using-mysql.html>

You need to provide these paths in the Extract and Replicat parameter files using the SETENV parameter.

Following are the SETENV environment parameters to set the two-way SSL connection:

- OGG_MYSQL_OPT_SSL_CA: Sets the full path of the certification authority.
- OGG_MYSQL_OPT_SSL_CERT: Sets the full path of the client certificate.
- OGG_MYSQL_OPT_SSL_KEY: Sets the full path of the client key.

In the following example, the MySQL SSL certificate authority, client certificate, and client key paths are set to the Oracle GoldenGate MySQL Extract and Replicat parameter:

```
SETENV (OGG_MYSQL_OPT_SSL_CA='/var/lib/mysql.pem')
SETENV (OGG_MYSQL_OPT_SSL_CERT='/var/lib/mysql/client-cert.pem')
SETENV (OGG_MYSQL_OPT_SSL_KEY='/var/lib/mysql/client-key.pem')
```

For a MySQL user configured with X509 encryption scheme, the MySQL database requires the ssl-key and ssl-cert options at the time of logging in. So, when an Oracle GoldenGate credential store entry is created for this user, the SSL options in the credential store alias must mandatorily include sslKey and sslCert regardless of sslMode used.

Database Configuration

Learn about supported MySQL databases, required settings, and how to prepare tables for processing as part of configuring MySQL for Oracle GoldenGate.

Supported Databases

Oracle GoldenGate for MySQL supports capture and delivery for MySQL, Oracle MySQL Heatwave, Amazon Aurora MySQL, Amazon RDS for MariaDB, Amazon RDS for MySQL, Azure Database for MySQL, Google Cloud SQL for MySQL, and MariaDB.

Oracle GoldenGate supports delivery to SingleStoreDB and SingleStoreDB Cloud, using the Oracle GoldenGate for MySQL Replicat.

Capture and delivery for MySQL configured with Group Replication in single-primary mode is supported. For more information, see [Using Oracle GoldenGate with MySQL Group Replication](#).

For a complete list of supported databases and versions, review the [Certification Matrix](#) for your version of Oracle GoldenGate.

Limitations of Support

The following are the limitations of support for Oracle GoldenGate for MySQL:

- MySQL databases enabled with binary log transaction compression are not supported with Oracle GoldenGate Extract.
- MySQL databases enabled with binary log encryption are not supported with Oracle GoldenGate Extract.

Database Storage Engine

Requirements for the database storage engine are as follows:

- Oracle GoldenGate supports the InnoDB storage engine for a source MySQL database.
- Oracle GoldenGate supports capture and apply from and to the InnoDB engine. Apply to MyISAM engine works, but there might be data integrity issues as MyISAM engine is non-transactional.

Database Character Set

MySQL provides a facility that allows users to specify different character sets at different levels.

Level	Example
Database	<code>create database test charset utf8;</code>
Table	<code>create table test(id int, name char(100)) charset utf8;</code>
Column	<code>create table test (id int, name1 char(100) charset gbk, name2 char(100) charset utf8));</code>

Limitations of Support

- Binary collations are not supported for multi-byte character sets. For example, do not set the `collation_server` variable equal to `utf8mb4_bin` when the character set is `utf8mb4`.
- The following character sets are **not** supported:
 - `armscii8`
 - `geostd8`
 - `keybcs2`
 - `utf16le`

Set the Session Character Set

The Extract and Replicat processes use a session character set when connecting to the database from the command line interface (Admin Client). For MySQL, the session character set is taken from the `SESSIONCHARSET` option of the `SOURCEDB` and the `TARGETDB` parameters.

Make certain that you specify a session character set in one of these ways when you configure Oracle GoldenGate.

Configure MySQL for Remote Capture

Oracle GoldenGate remote capture for MySQL, Amazon RDS for MySQL, Amazon Aurora MySQL, Azure Database for MySQL is used to capture transaction log data from a database located remotely to the Oracle GoldenGate installation.

Database Server Configuration

For remote capture to work, configure the MySQL server as follows:

1. Grant access permissions to the Oracle GoldenGate remote capture user.

Run the following statements against the remote database to create the user and grant the permissions needed for remote capture.

```
CREATE USER 'username'@'host' IDENTIFIED BY 'Password';  
GRANT ALL PRIVILEGES ON *.* TO 'username'@'host' WITH GRANT OPTION;  
FLUSH PRIVILEGES;
```

2. The `server_id` value of the remote MySQL server should be greater than 0. This value can be verified by issuing the following statement on the MySQL remote server:

```
SHOW VARIABLES LIKE 'server_id';
```

If the `server_id` value is 0, modify the `my.cnf` configuration file to set to a value greater than 0.

Oracle GoldenGate Configuration

Oracle GoldenGate configuration has the following steps:

1. Provide the remote database's connection information in the Extract's parameter file.

```
SOURCEDB remotedb@mysqlserver.company.com:port, USERID username, PASSWORD  
password
```

2. Add the following parameter to the Extract's parameter file, after the connection information.

```
TRANLOGOPTIONS ALTLOGDEST REMOTE
```

Limitations of Oracle GoldenGate Remote Capture for MySQL

Co-existence of Oracle GoldenGate for MySQL remote capture with the MySQL's native replication slave is supported with following conditions and limitations:

- Oracle GoldenGate for MySQL remote capture does not support an update operation that results in the size of before and after image exceeding 1 GB.

In an update operation, when the combined size of before and after image exceeds 1 GB, the remote capture API gets the following error from the MySQL server:

```
log event entry exceeded max_allowed_packet
```

- The native replication slave should be assigned a different `server_id` than the currently running slaves. The slave `server_id` values can be seen using the following MySQL command on the master server.

```
SHOW SLAVE HOSTS;
```

- If the Oracle GoldenGate capture abends with error "A slave with the same `server_uuid` or `server_id` as this slave has connected to the master", then change the capture's name and restart the capture.
- If the native replication slave dies with the error "A slave with the same `server_uuid` or `server_id` as this slave has connected to the master", then change the native replication slave's `server_id` and restart it.

- Remote capture is supported for Oracle GoldenGate on running on Linux and can support databases running on Linux or Windows.

Transaction Log Settings and Requirements

Learn how to prepare MySQL database and supported variants for Oracle GoldenGate, including required binary logging setting and database server variables.

Configuring Logging Properties

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

This section addresses the following binary log startup options and system variables that apply to Oracle GoldenGate for MySQL.

Logging Options	Values	What It Does	Requirement
binlog_annotate_row_events	ON, OFF	System variable that tells the primary server to write these events to the binary log.	Required for MariaDB and Oracle GoldenGate 23.8 and prior. OFF
binlog_expire_logs_seconds	0 - 4294967295	Where supported, this system variable sets the binary log expiration period, in seconds.	Recommended value is: ≥ 86400
binlog_format	ROW, STATEMENT, MIXED	System variable that sets the binary logging format.	Required when MySQL is used as source database. If set to MIXED or STATEMENT, only ROW level transactions will be captured. ROW
binlog retention hours	1 - 168	Where supported, this system variable sets the binary log expiration period, in hours.	Recommended value is: ≥ 24
binlog_row_image	FULL, MINIMAL, NOBLOB	System variable that sets different levels of column level logging in the binary log.	Required when MySQL is used as source database. FULL
binlog_row_metadata	FULL, MINIMAL	System variable that configures the amount of metadata added to the binary log for row-based logging.	Required when MySQL 8.0 or above is used as source database for DDL support. FULL
event_scheduler	ON, OFF	System variable that allows creation of Heartbeat update and purge jobs.	Required if implementing the Oracle GoldenGate Heartbeat functionality. ON

Logging Options	Values	What It Does	Requirement
log_bin	ON, OFF	System variable that shows the status of binary logging.	Required when MySQL is used as source database. If not already ON, set by assigning a name, such as log-bin=binlog.
log_replica_updates	ON, OFF	System variable that logs updates into the secondary server's own binary log when received from a primary server.	Required when MySQL is used as source database and is configured to capture from a replica server. ON
server_id	0 - 4294967295	System variable required to be specified when binary logging is enabled. For a replication primary and each replication replica, ensure values are >0 and unique.	Required when MySQL is used as source database. >=1

Verify Database-Level Logging

To verify the database's current binary logging settings required to support Oracle GoldenGate Capture for the MySQL database, perform the following queries. If settings need to be change, follow the instructions under [Enable Database-level Logging](#)

1. Log in to MySQL as the root or assigned database service Administrator account.

```
[root@srcdbserver ~]$ mysql -uroot -p
```

2. Issue the following command to determine whether the database is enabled for binary logging. If the result is ON, then database binary log is enabled.

```
show variables like 'log_bin';
```

3. Issue the following commands to determine whether the database is enabled with the correct binary log settings and server_id. Ensure that the binlog_format value is set to ROW, the binlog_row_image and binlog_row_metadata are set to FULL, and that the server_id variable is set to >=1 and are unique for a primary and any replicas.

```
show variables like 'binlog_%';
show variables like 'server_id';
```

4. Repeat the process to query the variables for the other options listed in the [Configuring Logging Properties](#), based on your requirements, such as DDL support and MariaDB support.

Enable Database-level Logging

Perform the following steps, if necessary, to enable the database binary logging settings required to support Oracle GoldenGate Capture for a MySQL database. Only modify those settings that need to be enabled or changed, based on the results from [Verify Database-Level Logging](#).

1. Edit the MySQL configuration file, `my.cnf` for Linux, or `my.ini` for Windows, to enable system variables that are required to support Oracle GoldenGate capture for MySQL.

```
[root@srcdbserver ~]$ vi /etc/my.cnf
```

2. To enable the binary log, add an entry to the configuration file, like the following, using a `log-bin` value of your preference.

```
log-bin=binlog
```

3. To set the correct binary logging format and `server_id` variables, set the following:

```
binlog-format=ROW
```

```
binlog-row-image=FULL
```

```
#For MySQL 8.0 and above, for DDL support
```

```
binlog-row-metadata=FULL
```

```
#Set to value > 0 and <= 4294967295
```

```
server-id=1
```

4. Add additional variables for the other options listed in [Configuring Logging Properties](#), based on your requirements, such as DDL support and MariaDB support.
5. Save and close the file.
6. Restart the database for the settings to take effect.

```
[root@srcdbserver ~]$ service mysqld restart
```

7. Check the status of the database service using the following command:

```
[root@srcdbserver ~]$ service mysqld status
```

Modifying System Variables for Amazon and Azure MySQL Database Services

To configure Amazon and Azure MySQL databases for Oracle GoldenGate Extract and heartbeat functionality, required changes must be done using the parameter groups, database creation or modification, or server parameter settings, if not already set to the required values. Use the following guidelines to modify these system variables:

1. To enable binary logging in Amazon, you must enable backups of the database, otherwise the `log_bin` variable will report as OFF and Extract will abend, reporting that it cannot retrieve the list of binlog files.

Enable backups at database creation or by modify an existing database to include a backup regiment.

2. For Amazon MySQL databases, set the `binlog_format` variable to ROW, the `binlog_row_image` variable to FULL, the `event_scheduler` to ON if implementing the Oracle GoldenGate heartbeat functionality, and set the `binlog_row_metadata` variable to FULL if enabling DDL capture support.

Follow the instructions provided by Amazon to create or edit an existing Parameter group in order to make these changes.

<https://aws.amazon.com/premiumsupport/knowledge-center/enable-binary-logging-aurora/>

3. For Azure MySQL databases, set the `binlog_format` variable to `ROW` and the `binlog_row_image` variable to `FULL`. Follow the instructions provided by Microsoft to modify the server parameter settings for the database instance.

<https://docs.microsoft.com/en-us/azure/mysql/howto-server-parameters>

Ensuring Data Availability

Retain enough binary log data so that if you stop Extract or there is an unplanned outage, Extract can start again from its checkpoints. Extract must have access to the binary log that contains the start of the oldest uncommitted unit of work, and all binary logs thereafter. The recommended retention period is at least 24 hours worth of transaction data, including both active and archived information. You might need to do some testing to determine the best retention time given your data volume and business requirements.

If data that Extract needs during processing was not retained, either in active or backup logs, one of the following corrective actions might be required:

- Alter Extract to capture from a later point in time for which binary log data is available (and accept possible data loss on the target).
- Resynchronize the source and target tables, and then start the Oracle GoldenGate environment over again.

Binary log data retention can be increased with different measures depending on the MySQL database or variant.

- For MySQL Server, MySQL Heatwave, Azure Database for MySQL, and MariaDB, the variable `binlog_expire_logs_seconds` can be increased to adjust the retention period.
- For Amazon Aurora for MySQL and Amazon RDS for MySQL, the default binary log retention settings are `NULL`, which means, for Aurora the logs are lazily purged and for RDS for MySQL, they are not retained. Use the following procedure to set the parameter `binlog retention hours`. For example:

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

Capturing Against a MySQL Replication Replica

You can configure Oracle GoldenGate to capture from a MySQL replication replica.

Typically, the transactions applied at the replica are logged into the relay logs and not into the replica's binlog. For the replica to write transactions that it receives from the primary, into its binlog, you must start the replication replica with the `log_replica_updates` option as 1 in `my.cnf` in conjunction with the other binary logging parameters for Oracle GoldenGate. After the primary's transactions are in the replica's binlog, you can set up a regular Oracle GoldenGate Extract on the replica to begin capture.

Prepare Tables for Processing

This section describes how to prepare the tables for processing. Table preparation requires the following tasks.

Ensure Row Uniqueness for Tables

Oracle GoldenGate requires some form of 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:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. 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 those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration.

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.

4. If a table does not have an appropriate key, or if you prefer that the existing key(s) are not 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. See TABLE | MAP in *Reference for Oracle GoldenGate*.

Tables with a Primary Key Derived from a Unique Index

In the absence of a primary key on a table, MySQL will promote a unique index to primary key if the indexed column is NOT NULL. If there are more than one of these not-null indexes, the first one that was created becomes the primary key. To avoid Replicat errors, create these indexes in the same order on the source and target tables.

For example, assume that source and target tables named ggvam.emp each have columns named first, middle, and last, and all are defined as NOT NULL. If you create unique indexes in the following order, Oracle GoldenGate will abend on the target because the table definitions do not match.

Source:

```
CREATE UNIQUE INDEX UQL ON ggvam.emp(first);  
CREATE UNIQUE INDEX UQ2 on ggvam.emp(middle);  
CREATE UNIQUE INDEX UQ3 on ggvam.emp(last);
```

Target:

```
CREATE UNIQUE INDEX UQ1 ON ggvam.emp(last);  
CREATE UNIQUE INDEX UQ2 ON ggvam.emp(first);  
CREATE UNIQUE INDEX UQ3 ON ggvam.emp(middle);
```

The result of this sequence is that MySQL promotes the index on the source "first" column to primary key, and it promotes the index on the target "last" column to primary key. Oracle

GoldenGate will select the primary keys as identifiers when it builds its metadata record, and the metadata will not match. To avoid this error, decide which column you want to promote to primary key, and create that index first on the source and target.

Specify Your Own Key for Oracle GoldenGate to Use

If a table does not have one of the preceding types of row identifiers, or if you prefer those identifiers 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.

Limit Row Changes in Tables That Do Not Have a Key

If a target table does not have a primary key or a unique key, duplicate rows can exist. In this case, Oracle GoldenGate could update or delete too many target rows, causing the source and target data to go out of synchronization without error messages to alert you. To limit the number of rows that are updated, use the DBOPTIONS parameter with the LIMITROWS option in the Replicat parameter file. LIMITROWS can increase the performance of Oracle GoldenGate on the target system because only one row is processed.

Triggers and Cascade Constraints Considerations

Triggers

Disable triggers on the target tables, or alter them to ignore changes made by the Oracle GoldenGate database user. Oracle GoldenGate replicates DML that results from a trigger. If the same trigger gets activated on the target table, then it becomes redundant because of the replicated version, and the database returns an error.

Cascade Constraints Considerations

Cascading updates and deletes captured by Oracle GoldenGate are not logged in binary log, so they are not captured. This is valid for both MySQL and MariaDB. For example, when you run the delete statement in the parent table with a parent child relationship between tables, the cascading deletes (if there are any) happens for child table, but they are not logged in binary log. Only the delete or update record for the parent table is logged in the binary log and captured by Oracle GoldenGate.

See <https://mariadb.com/kb/en/replication-and-foreign-keys/> and <https://dev.mysql.com/doc/refman/8.0/en/innodb-and-mysql-replication.html> for details.

To properly handle replication of cascading operations, it is recommended to disable cascade deletes and updates on the source and code your application to explicitly delete or update the child records prior to modifying the parent record. Alternatively, you must ensure that the target parent table has the same cascade constraints configured as the source parent table, but this could lead to an out-of-sync condition between source and target, especially in cases of bi-directional replication.

MySQL: Supported Data Types, Objects, and Operations

This section contains support information for Oracle GoldenGate on MySQL Database.

Oracle GoldenGate for MySQL supports the mapping, filtering, and transformation of source data, unless noted otherwise in this document, along with replicating data derived from other source databases supported by Oracle GoldenGate, into MySQL databases.

Oracle GoldenGate for MySQL Supported Data Types

Oracle GoldenGate for MySQL supports the following data types:

- BLOB
- BIGINT
- BINARY
- BIT(M)
- CHAR
- DATE
- DATETIME
- DECIMAL
- DOUBLE
- ENUM
- FLOAT
- INT
- JSON
- LONGBLOB
- LONGTEXT
- MEDIUMBLOB
- MEDIUMINT
- MEDIUMTEXT
- SMALLINT
- TEXT
- TIME
- TIMESTAMP
- TINYBLOB
- TINYINT
- TINYTEXT
- VARBINARY
- VARCHAR
- VECTOR
- YEAR

Limitations and Clarifications

When running Oracle GoldenGate for MySQL, be aware of the following:

- Functional indexes are not supported for Capture or Delivery.
- Oracle GoldenGate does not support BLOB or TEXT types when used as a primary key.

- Oracle GoldenGate supports a TIME type range from 00:00:00 to 23:59:59.
- 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 time zone, conversion may add or subtract hours, which can cause the timestamp to exceed the lower or upper supported limit.
- Oracle GoldenGate does not support negative dates.
- The support of 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.
- When you use ENUM type in non-strict sql_mode, the non-strict sql_mode does not prevent you from entering an invalid ENUM value and an error will be returned. To avoid this situation, do one of the following:
 - Use sql_mode as STRICT and restart Extract. This prevents users from entering invalid values for any of the data types. An IE user can only enter valid values for those data types.
 - Continue using non-strict sql_mode, but do not use ENUM data types.
 - Continue using non-strict sql_mode and use ENUM data types with valid values in the database. If you specify invalid values, the database will silently accept them and Extract willabend.
- Table with single column is not supported for JSON datatype. Extract willabend in case it is configured for a table which has a single column of JSON datatype.
- JSON datatype does not support CDR. The following message gets logged in the report file if GETBEFORECOLS is configured and the table has columns of JSON datatypes:

INFO 0GG-06556 The following columns will not be considered for CDR

The limitations for CDR applies to cases where the GETBEFORECOLS and COMPARECOLS are used.

- The MySQL Extract does notabend and processes the ENUM column values that are not defined in the table definition as 'invalid_enum'.

Non-Supported MySQL Data Types

Oracle GoldenGate for MySQL does not support the following data types:

All spatial types (GEOMETRY and so on), and SET.

All the following GEOMETRY types are unsupported in MySQL:

- POINT
- LINESTRING
- POLYGON
- MULTIPOINT
- MULTILINESTRING
- MULTIPOLYGON
- GEOMCOLLECTION

Note

Extract abends if it is configured to capture from tables that contain any of the unsupported data types, so ensure that Extract is not configured to capture from tables containing columns of unsupported data types.

Supported Objects and Operations for MySQL

Oracle GoldenGate for MySQL supports the following objects and operations:

- Oracle GoldenGate supports the following DML operations on source and target database transactional tables:
 - Insert operation
 - Update operation (compressed included)
 - Delete operation (compressed included)
 - Truncate operation
- Oracle GoldenGate supports the extraction and replication of DDL (data definition language) operations.
- Oracle GoldenGate supports transactional tables up to the full row size and maximum number of columns that are supported by MySQL and the database storage engine that is being used. InnoDB supports up to 1017 columns.
- Generated columns are supported and captured.
- Oracle GoldenGate supports the AUTO_INCREMENT column attribute. The increment value is captured from the binary log by Extract and applied to the target table in a Replicat insert operation.
- Oracle GoldenGate can operate concurrently with MySQL native replication.
- Oracle GoldenGate supports the DYNSQL feature for MySQL.

Note

XA transactions are not supported for capture and any XA transactions logged in binlog cause Extract to abend. So, you must not use XA transactions against a database that Extract is configured to capture.

If XA transactions are being used for databases that are not configured for Oracle GoldenGate capture, then exclude those databases from logging into MySQL binary logs by using the parameter `binlog-ignore-db` in the MySQL server configuration file.

Limitations on Automatic Heartbeat Table support are as follows:

- Ensure that the database in which the heartbeat table is to be created already exists to avoid errors when adding the heartbeat table.
- In the heartbeat history lag view, the information in fields like `heartbeat_received_ts`, `incoming_heartbeat_age`, and `outgoing_heartbeat_age` are shown with respect to the system time. You should ensure that the operating system time is setup with the correct and current time zone information.

- Position by End of File (EOF) is supported in MySQL. Oracle GoldenGate Extract for MySQL finds the position corresponding to the end of the file and starts reading transactions from there. The EOF position is not exact, if data is continuously written to the binary log.

The Extract is added and altered using:

```
ADD EXTRACT group_name, TRANLOG, EOF
```

```
ALTER EXTRACT group_name, EOF
```

Details of Support for Objects and Operations in MySQL DDL

Here's a list of the MySQL objects and operation types that Oracle GoldenGate supports for the capture and replication of DDL operations.

- DDL replication for MySQL is only supported between MySQL databases as sources and targets.
- Extraction and replication of DDL operations are supported for MySQL 8.0 and higher. The supported DDL operations include:
 - CREATE TABLE (excluding CREATE TABLE AS SELECT)
 - DROP TABLE
 - TRUNCATE TABLE (Support using GETTRUNCATES)
 - ALTER TABLE TRUNCATE PARTITION
 - ALTER TABLE ADD COLUMN
 - ALTER TABLE ADD COLUMN AFTER
 - ALTER TABLE DROP COLUMN
 - ALTER TABLE MODIFY COLUMN (COLUMN_NAME, TYPE, SIZE)
 - ALTER TABLE ADD (*constraints*) PRIMARY KEY
 - ALTER TABLE DROP PRIMARY KEY
- For MySQL 8.0, local and remote DDL capture is supported.
- TRUNCATE operations are supported as DML through the GETTRUNCATES Extract and Replicat parameter and do not require configuring Oracle GoldenGate for MySQL DDL support.
- DDL replication is not supported in a Oracle GoldenGate bi-directional configuration.
- DDL replication is not supported for cloud based database services where the `binlog_row_metadata` database setting cannot be set to FULL.

Non-Supported Objects and Operations for MySQL

Oracle GoldenGate for MySQL does not support the following objects and operations:

- Invisible columns
- The Oracle GoldenGate BATCHSQL feature for Replicat.
- The following character sets are not supported:

```
ULIB_CS_ARMSCII8, /* American National 166-9 */
```

```
ULIB_CS_GEOSTD8, /* Georgian Standard */  
ULIB_CS_KEYBCS2, /* Kemennicky MS-DOS
```

- Capturing NLS LOB data using the `FETCHMOCOLS` and `FETCHMODCOLEXCEPT TABLE` options is not supported when DDL is enabled.
- Renaming tables.
- DDL statements inside stored procedures is not supported.
- When the time zone of the Oracle GoldenGate installation server does not match the time zone of the source database server, then the `TIMESTAMP` data sent to the target database will differ from the source database. For Oracle GoldenGate Microservices installations, regardless of the time zones being the same, Extract will resolve the time zone to UTC. Determine the source database time zone by running the following query:

```
select @@system_time_zone;
```

This will return a time zone value, such as PDT.

Create a variable in the deployment that contains the source Extract, called `TZ` and set it to the value of the source database time zone. After this, stop any running Oracle GoldenGate processes and restart the Administration Service, and then start the Extracts and Replicats.

- Extraction and replication from and to views is not supported.
- Transactions applied by the slave are logged into the relay logs and not into the slave's binlog. If you want a slave to write transactions the binlog that it receives from the master, you need to start the replication slave with the `log slave-updates` option as 1 in `my.cnf`. This is in addition to the other binary logging parameters. After the master's transactions are in the slave's binlog, you can then setup a regular capture on the slave to capture and process the slave's binlog.

Limitations of SingleStoreDB Objects and Operations

Oracle GoldenGate for MySQL now supports SingleStoreDB and SingleStoreDB Cloud, but with the following limitations:

- Replication to views within SingleStoreDB are not supported, as only the base tables are writeable.
- Updates to columns that are part of the SHARD key are not supported.
- Primary key updates on tables with no explicit SHARD key are not supported. This is because SingleStoreDB assigns the primary key as the SHARD key in this situation, and updates to SHARD key columns are not allowed.
- The `DBOPTIONS LIMITROWS` behavior of Replicat for SingleStoreDB tables without a primary or unique key that are spread across multiple partitions, is not supported.
- The `DBOPTIONS LIMITROWS` and `NOLIMITROWS` parameter options are not supported for SingleStoreDB.
- SingleStoreDB does not support cross-database transactions, which means that a Replicat can only support mapping to a single schema/database. This includes mappings for checkpoint and heartbeat tables, so these objects must be created under the same schema/database as the user tables to be replicated.

System Schemas

The following schemas or objects are not automatically replicated by Oracle GoldenGate unless they are explicitly specified without a wildcard.

- MySQL
 - 'information_schema'
 - 'performance_schema'
 - 'mysql'
 - 'sys'
- SingleStore
 - 'information_schema'
 - 'cluster'
 - 'memsql'

Oracle

Prepare your database for Oracle GoldenGate, including configuring connections and logging, enabling Oracle GoldenGate in your database, setting up the flashback query, and managing server resources.

Prepare Database Users and Privileges for Oracle

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

Note

Both Extract and Replicat require the database to be in Read/Write mode. An Extract cannot capture from a Data Guard standby database, or Active Data Guard standby database that is open for read-only activity, even with DML redirection enabled.

Grant User Privileges for Oracle AI Database 26ai and Higher

Oracle AI Database 26ai provides user roles to grant role-based privileges to a database user with SQL statements.

Oracle AI Database 26ai uses a role-based approach to grant privileges necessary for replication. The roles are a replacement of the GRANT_ADMIN_PRIVILEGE procedures from the DBMS_GOLDENGATE_AUTH package.

For example, to create an Oracle GoldenGate Administration user, you have to grant the appropriate OGG_* role to the user while creating the user. If you still use GRANT_ADMIN_PRIVILEGE procedures from the DBMS_GOLDENGATE_AUTH package in Oracle AI Database 26ai, no privileges are granted. Instead, a warning message is raised alerting that the procedure call is disabled.

Here are the user roles introduced in Oracle GoldenGate 23ai:

OGG_CAPTURE

OGG_CAPTURE has the privileges necessary for using and managing Extract processes. For capturing DML and DDL with Extract, a user requires the following permissions:

```
GRANT CONNECT, RESOURCE to ggadmin;  
GRANT OGG_CAPTURE to ggadmin;
```

If the Extract process is registered at the root container, then this additional step must be taken:

```
ALTER USER c##ggadmin SET CONTAINER_DATA = all CONTAINER = current;
```

OGG_APPLY

Role with privileges necessary for using Oracle GoldenGate Replicat.

To use a Replicat processes a user needs this role, as well as the permissions to execute DML and DDL at the target. For example, if a Replicat process is intended to perform DML operations on the EMPLOYEES table from the HR schema:

```
GRANT CONNECT, RESOURCE to ggadmin;  
GRANT OGG_APPLY to ggadmin;  
GRANT SELECT, INSERT, UPDATE, DELETE on HR.EMPLOYEES TO GGADMIN;
```

If the Replicat user is intended to apply DDL operations like CREATE TABLE, DROP TABLE and ALTER TABLE it should receive the system privileges necessary to execute such statements:

```
GRANT CREATE TABLE, ALTER TABLE, DROP TABLE to ggadmin;
```

OGG_APPLY_PROCREP

Role with privileges necessary to execute packages supported for procedural replication with Oracle GoldenGate. It only includes the execution permissions, therefore, this role should be used together with OGG_APPLY role to allow the user to run the Replicat process and to execute the procedures at the target.

For example, if a Replicat will apply procedure executions. The grant process should be as follows:

```
GRANT CONNECT, RESOURCE to ggadmin;  
GRANT OGG_APPLY, OGG_APPLY_PROCREP to ggadmin;
```

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	Required to create objects In Oracle Database 12cR1 and later, instead of RESOURCE, grant the following privilege: ALTER USER user QUOTA {size UNLIMITED} ON tablespace;
CONNECT	Yes	Yes	Common user SYSTEM connects to the root container. This privilege is essential when the DBA role is not assigned to the user. See an example of Permissions granted to an Oracle multitenant database common user.
CREATE PROCEDURE	Yes	Yes	Required to add heartbeat tables.
CREATE SESSION	Yes	Yes	Required to connect to the database.
CREATE VIEW	Yes	Yes	Required to add the heartbeat table view. 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.
ALTER SYSTEM	Yes	Yes	Perform administrative changes, such as enabling logging.
ALTER USER	Yes	Yes	Required for multitenant architecture and <code>GGADMIN</code> should be a valid Oracle GoldenGate administrator schema.

Privilege	Extract	Replicat All Modes	Purpose
EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE ('REPUSER', CONTAINER=>'PDBEAST');	Yes	Yes	- Required for Autonomous AI 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
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 .

Privilege	Extract	Replicat All Modes	Purpose
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.

Privileges for Capturing from Oracle Data Vault

Grant the following privileges connected as SYS user in Oracle AI 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');
```

Prepare Database Connection, System, and Parameter Settings

Learn about configuring database connection, system, and parameter settings for Oracle GoldenGate for Oracle.

Enable GoldenGate Replication and Archive Mode

Step 1: Enable GoldenGate Replication

The database services required to support Oracle GoldenGate Extract and Replicat must be enabled explicitly for Oracle database.

To enable Oracle GoldenGate replication, set the following database initialization parameter. All instances in Oracle RAC must have this value set to TRUE if using Oracle GoldenGate on any of the nodes.

```
ENABLE_GOLDENGATE_REPLICATION=true
```

For more information about this parameter, see Initialization Parameters.

Steps 2: Enable the Archive Mode

Oracle Databases must be in ARCHIVELOG mode so that Extract can process the log files. To switch on the ARCHIVELOG mode, follow the steps provided in Changing the Database Archiving Mode.

Setting Flashback Query

To know about the data that Oracle GoldenGate fetches, see [Details of Support for Oracle Data Types and 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.

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.

Handling Other Database Properties

There are some database properties that may affect Oracle GoldenGate and the parameters used to resolve or work around certain conditions.

The following table lists the database properties and the associated concern/resolution.

Database Property	Concern/Resolution
Table with interval partitioning	To support tables with interval partitioning, make certain that the <code>WILDCARDRESOLVE</code> parameter remains at its default of <code>DYNAMIC</code> .
Table with virtual columns	Virtual columns are not logged, and Oracle does not permit DML on virtual columns. You can, however, capture this data and map it to a target column that is not a virtual column by doing the following: Include the table in the Extract <code>TABLE</code> statement and use the <code>FETCHCOLS</code> option of <code>TABLE</code> to fetch the value from the virtual column in the database. In the Replicat <code>MAP</code> statement, map the source virtual column to the non-virtual target column.

Database Property	Concern/Resolution
Table with inherently updateable view	To replicate to an inherently updateable view, define a key on the unique columns in the updateable view by using a KEYCOLS clause in the same MAP statement in which the associated source and target tables are mapped.
Redo logs or archives in different locations	The TRANLOGOPTIONS parameter contains options to handle environments where the redo logs or archives are stored in a different location than the database default or on a different platform from that on which Extract is running.
TRUNCATE operations	To replicate TRUNCATE operations, choose one of two options: - Standalone TRUNCATE support by means of the GETTRUNCATES parameter replicates TRUNCATE TABLE, but no other TRUNCATE options. Use only if not using Oracle GoldenGate DDL support. - The full DDL support replicates TRUNCATE TABLE, ALTER TABLE TRUNCATE PARTITION, and other DDL.
Sequences	To replicate DDL for sequences (CREATE, ALTER, DROP, RENAME), use Oracle GoldenGate DDL support. To replicate just sequence values, use the SEQUENCE parameter in the Extract parameter file. This does <i>not</i> require the Oracle GoldenGate DDL support environment.

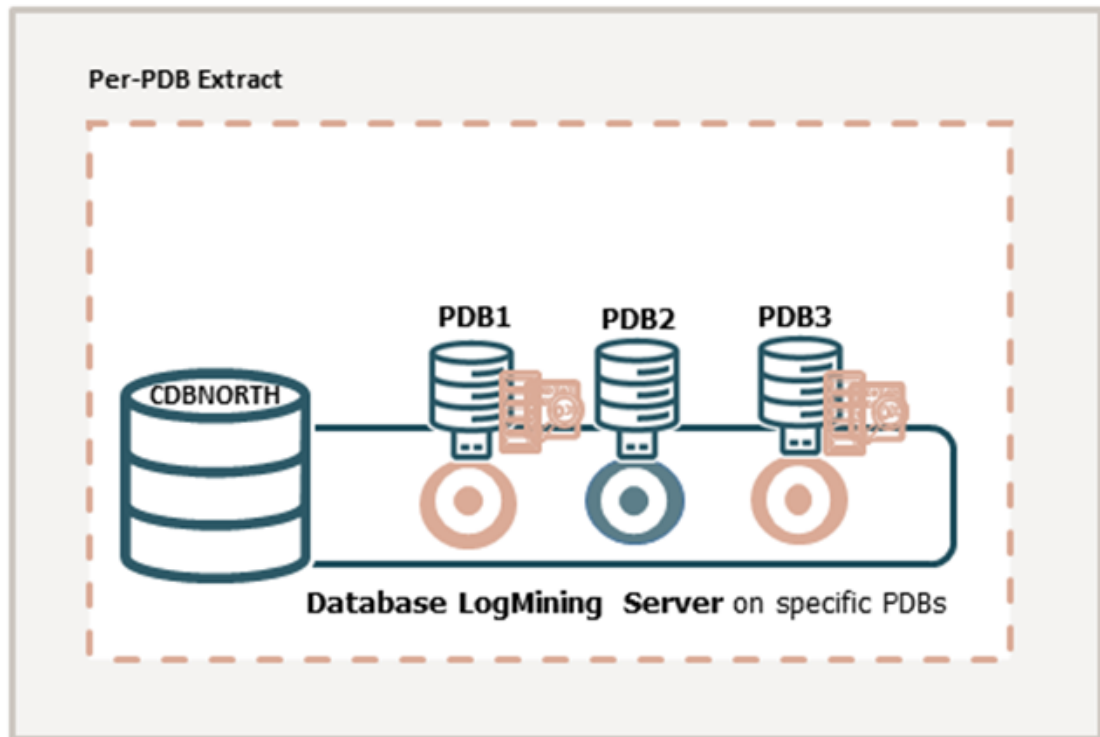
Configure a Multitenant Container Database

Oracle GoldenGate with Oracle AI Database allows the implementation of pluggable databases (PDBs) for source and target. Extract is registered for a specific PDB, which is called a per-PDB Extract.

Note

Root level access with the c##ggadmin user is not recommended, except in case of downstream Extract configuration. Use Extract at the per-PDB level instead of root level Extract.

The following diagram shows the configuration for different approaches in a multitenant container database configuration:



Adding Extract directly from the PDB captures from isolated PDBs, managing ownership and responsibility at the PDB level.

Using a per-PDB Extract, you can connect as the local PDB user (for example, ggadmin) and then register this Extract with the database. As you are already logged in as the PDB user, an additional container clause is *not* required. Similarly, the SOURCECATALOG or a three-part naming convention is also not needed.

To set up an Extract, see [Add an Online Extract](#).

Considerations for Multitenant Container Database Configuration

Consider the following guidelines when configuring a multitenant container databases for data replication using Oracle GoldenGate:

- 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.
- To create and register a per-PDB Extract, you will need to connect to the PDB user such as ggadmin created for PDB-level access. Use the USERIDALIAS parameter to configure a SQL*Net connection string such as ggadmin@pdbeast. You do *not* need the container clause or the SOURCECATALOG to set up the per-PDB Extract.
- If the PDB user name, CDB user name, or GLOBAL_NAME contains a dash in it, then Oracle GoldenGate throws the following error:

```
ERROR   OGG-08221  Cannot register or unregister EXTRACT <extract_name>
because of the
following SQL error: OCI Error ORA (status = 44004-ORA-44004: invalid
qualified SQL name
```


- 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.
- DDL replication works as a normal replication for multitenant databases.

See [Add Database Connections](#) to add a multitenant container database user in Oracle GoldenGate credentials. See [Grant User Privileges for Oracle Database 21c and Lower](#) or [Grant User Privileges for Oracle AI Database 26ai and Higher](#) depending on the Oracle database installation that you need to configure.

Flush Sequence for Multitenant Container Database

You can only use the `FLUSH SEQUENCE` command within Oracle GoldenGate, if the `sequence.sql` script applies the database procedures into the GoldenGate Admin schema of the database.

Use the `FLUSH SEQUENCE` command immediately after you start Extract for the first time during an initial synchronization or a re-synchronization. This command updates an Oracle sequence, so that initial redo records are available at the time that Extract starts to capture transaction data. Normally, redo is not generated until the current cache is exhausted. The flush gives Replicat an initial start point with which to synchronize to the correct sequence value on the target system. From then on, Extract can use the redo that is associated with the usual cache reservation of sequence values.

1. The following Oracle procedures are used by `FLUSH SEQUENCE`:

Database	Procedure	User and Privileges
Source	<code>updateSequence</code>	Grants EXECUTE to the owner of the Oracle GoldenGate DDL objects, or other selected user if not using DDL support.
Target	<code>replicateSequence</code>	Grants EXECUTE to the Oracle GoldenGate Replicat user.

The `sequence.sql` script installs these procedures. Normally, this script is run as part of the Oracle GoldenGate installation process, but make certain that was done before using `FLUSH SEQUENCE`. If `sequence.sql` was not run, the flush fails and an error message similar to the following is generated:

```
Cannot flush sequence {0}. Refer to the Oracle GoldenGate for Oracle
documentation for instructions on how to set up and run the sequence.sql
script. Error {1}.
```

2. Before using `FLUSH SEQUENCE`, connect to the database using the `DBLOGIN` command.

`FLUSH SEQUENCE` must be issued at the PDB level, to create an Oracle GoldenGate user in each PDB for which the sequence replication is required. Use `DBLOGIN` to log into that PDB, and run 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.

In the following example, the environment setup is for Oracle 21c to Oracle 21c Replication, with integrated Extract, parallel Replicat using Oracle GoldenGate 21c (21.3.0).

The following table lists the names of source and target CDB, PDBs, and their corresponding user credentials for connecting to the database.

Source CDB	Target CDB
NORTH	SOUTH
PDB Name: DBEAST	PDB Name: DBWEST
Common user: c##ggadmin	PDB User: ggadmin
PDB user for sequences: ggate	

```
sqlplus system/manager
ALTER SESSION SET CONTAINER=CERTMISSN;
CREATE USER ggate IDENTIFIED BY password DEFAULT TABLESPACE USERS TEMPORARY
TABLESPACE TEMP QUOTA UNLIMITED ON USERS CONTAINER=CURRENT;
```

Run @sequence.sql

```
sqlplus system/manager
ALTER SESSION SET CONTAINER=DBEAST;
@sequence.sql
```

When prompted enter the following:

GGADMIN

Run the FLUSH SEQUENCE command:

```
DBLOGIN USERIDALIAS ggeast DOMAIN OracleGoldenGate
FLUSH SEQUENCE DBEAST.HR.*
```

Target Oracle GoldenGate Configuration:

```
sqlplus system/manager
ALTER SESSION SET CONTAINER =PDBWEST;
@sequence.sql
```

When prompted, enter the PDB user name ggadmin.

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

Configure the Auto Capture Mode for Extract

The **auto capture** mode allows automatically capturing the tables that have been enabled for Oracle GoldenGate **auto capture**.

See How to Capture Supplemental Logging for Oracle GoldenGate in the *Oracle Database Utilities* guide.

Here are some benefits of using the auto capture mode:

- Easy to configure captured table set
- No requirement to update TABLE/TABLEEXCLUDE parameter

- No need to stop or restart Extract when captured table set changes

Enabling Auto Capture Mode for Extract

Enable the **auto capture** mode using TRANLOGOPTIONS:

```
TRANLOGOPTIONS ENABLE_AUTO_CAPTURE | DISABLE_AUTO_CAPTURE
```

When Extract is running in the auto capture mode, don't filter an LCR if the object is not part of exclusion list set by TABLE EXCLUDE parameter or any inclusion list set by TABLE parameter.

The LIST TABLES command shows the list of tables enabled for AUTO_CAPTURE.

Note

Auto capture is available from Oracle GoldenGate 21c with Oracle Database 19.18 data patch and higher. In case of database upgrade, any Extract which was registered prior to Oracle Database 19.18 cannot be converted to auto capture. Only new Extracts that are created after upgrading to Oracle Database 19.18 and later, can be converted to auto capture Extract.

See [DML Auto Capture](#) and [Details of Support for Objects and Operations in Oracle DDL](#) to know about the DML and DDL considerations.

Also see this article [Oracle GoldenGate 21c: Auto Capture of Tables](#) to learn more.

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.

When Automatic (Shared) Memory Management is enabled (recommended), there is no need to set the STREAMS_POOL_SIZE environment variable for Oracle Database. This is because no minimum memory is required for the Streams Pool. However, if Automatic (Shared) Memory Management is not enabled, then the Streams Pool can use up to 10% of the Shared Pool if STREAMS_POOL_SIZE is not specified and a root-level Extract (for Downstream Capture) or integrated Replicat (deprecated in Oracle GoldenGate 23ai) is in use.

If you wish to allocate memory from the STREAMS POOL explicitly, calculate the STREAMS_POOL_SIZE by adding 25% to the sum of the MAX_SGA_SIZE values for each process. For example, Downstream Capture uses the maximum size of 1GB for the MAX_SGA_SIZE parameter, by default. In this case, STREAMS_POOL_SIZE might be set to 1.25GB. If there were two additional integrated Replicats, each with a MAX_SGA_SIZE of 1GB, then the STREAMS_POOL_SIZE would need to be 3.75GB.

Note that Streams pool is also used by other components of the database (like Oracle Stream and Advanced Queuing), so make certain to take them into account while sizing the Streams pool for Oracle GoldenGate.

Also see [DOC ID 2998659.1](#) and the blog [GoldenGate: How to manage the Streams Pool in the Oracle Database?](#) to know more.

Support for Oracle Sequences

To support Oracle sequences, you must install some database procedures.

From the SQL prompt, run the script `$OGG_HOME/lib/sql/legacy/sequence.sql` on the source and target database as a DBA.

In a container database (CDB), connect as a local user with DBA privileges in the pluggable database (PDB).

In a non-CDB, connect as DBA for the database.

The Oracle GoldenGate Admin User does not necessarily need DBA privileges. However, the Oracle GoldenGate Admin User must have the `SELECT ANY DICTIONARY` and the `[CREATE | ALTER|DROP] ANY SEQUENCE` privileges in addition to the privileges granted by the `OGG_CAPTURE | OGG_APPLY` role for Oracle AI Database 26ai and higher. For earlier Oracle database versions, you can use the procedure call `DBMS_GOLDEN_GATE_AUTH.GRANT_ADMIN_PRIVILEGE`.

The following example shows how to login to a PDB as the system user and run the `sequence.sql` script:

```
sqlplus system/***@pdbeast
@sequence.sql
```

You will be prompted to provide the Oracle GoldenGate Admin User, such as `ggadmin`.

When the script successfully finishes, it returns the status for sequence replication:

```
STATUS OF SEQUENCE SUPPORT
-----
SUCCESSFUL installation of Oracle Sequence Replication support
```

Ensuring Row Uniqueness in Source and Target Table

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 [Transaction Log Settings and Requirements](#)).

Note

`KEYCOLS` clause is not supported for JSON Relation Duality Views (JSON DVs) and JSON Collection Tables (JCTs) because they have a single, predefined key (`_id`).

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. 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*.

In case the following criteria matches for Oracle Replicat, you need to use unique index that matches to 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.

Transaction Log Settings and Requirements

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. The logging level that you use depends on Oracle GoldenGate 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	Command Name	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. FORCE LOGGING overrides any table-level NOLOGGING settings.
Minimum database-level supplemental logging See Enable Subset Database Replication 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	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> . This setting also ensures that any new object created in this schema will have the appropriate supplemental logging.	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 ALLCOLS.
Schema-level supplemental logging with unconditional logging for all supported columns. (See Enable Schema-level Supplemental Logging for non-supported column types.)	ADD SCHEMATRANDATA with ALLCOLS option	Enables unconditional supplemental logging of all of the columns in a table, for all of the tables in a schema. This setting also ensures that any new object created in this schema will have the appropriate supplemental logging.	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 HANDLECOLLISIONS parameter for initial loads.

Logging option	Command Name	What it does	Use case
Schema-level supplemental logging, minimal setting	ADD SCHEMATRANDATA with NOSCHEDULINGCOLS option	Enables unconditional supplemental logging of the primary key and all valid unique indexes of all tables in a schema. This setting also ensures that any new object created in this schema will have the appropriate supplemental logging.	Use only for non-integrated Replicat. This is the minimum required schema-level logging.
Table-level supplemental logging with built-in support for integrated Replicat See Enable 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 Enable 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 non-integrated Replicat and non-parallel Replicat. This is the minimum required table-level logging.
Table-level supplemental logging for JSON Relational Duality Views and JSON Collection Tables.	ADD TRANDATA with Duality View name and Collection Table name	Enables supplemental logging for the specified Duality Views and Collection Tables.	See Enable Supplemental Logging for JSON Relational Duality Views and JSON Collection Tables for guidelines. Also see, Details of Support for JSON-Relational Duality Views and JSON Collection Tables to know more.

Enable Subset Database Replication 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.

There is a fine-granular database supplemental logging mode called Subset Database Replication available in LogMiner, which is the basic recommended mode for all Oracle GoldenGate and XStream clients. It replaces the previously used Minimum Supplemental Logging mode.

The subset database replication logging is enabled at CDB\$ROOT (and all user-PDBs inherit it) currently.

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, subset database replication 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:

```
ALTER PLUGGABLE DATABASE pdbname ADD SUPPLEMENTAL LOG DATA SUBSET DATABASE  
REPLICATION;;  
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.

```
ALTER SYSTEM SWITCH LOGFILE;
```


Enable Schema-level Supplemental Logging

Oracle GoldenGate supports schema-level supplemental logging. Schema-level logging is required for an Oracle source and target databases when using the Oracle GoldenGate DDL replication feature, or on the source database if DDL can be performed (or executed) on the tables being replicated. In all other use cases, it is optional, but then you must use table-level logging instead (see [Enable 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.

- 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
```

Enable 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 [Enable 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.
- To enable supplemental logging for JSON Relational Duality Views (JSON DVs) and JSON Collection Tables, you need to follow the guidelines in [Enable Supplemental Logging for JSON Relational Duality Views and JSON Collection Tables](#).

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](#) 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.
 - 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 Table](#) 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.

Enable Supplemental Logging for JSON Relational Duality Views and JSON Collection Tables

Before replicating JSON Relational Duality Views and JSON Collection Tables, you need to set up supplemental logging in Oracle GoldenGate. By default, supplemental logging is not enabled for JSON Relational Duality Views and JSON Collection Tables. You can use the following guidelines when enabling supplemental logging for Duality Views and Collection Tables:

- **Enable Logical Replication:** Enabling logical replication (using `ADD TRANDATA`) is mandatory for capturing changes made to Duality Views and Collection Tables. This process enables the supplemental logging needed to track changes at the JSON document level.

See `ADD TRANDATA` for details.

On the Oracle AI Database side, a clause has been added in the `CREATE/ALTER DDL` for JSON Relational Duality Views and JSON Collection Tables. This clause enables supplemental logging at the view level. This logging is in addition to any existing logging on the base tables.

For details, see the *logical_replication_clause* section in `CREATE TABLE/CREATE JSON RELATIONAL DUALITY VIEW` and `ALTER TABLE/ALTER JSON RELATIONAL DUALITY VIEWS` commands in *Oracle Database SQL Language Reference*.

- **DML changes for JSON Duality Views and JSON Collection Tables:** For `INSERT` and `UPDATE` operations, the full JSON document and the `_id` key attribute is supplementally logged. For `DELETE` operations, only the key `_id` attribute is logged.
- **Transaction Commit Logging for JSON Relational Duality Views:** Supplemental logging for Duality Views occurs at commit time. Only the Net changes are logged.

For example, if you `INSERT` a record and `UPDATE` this record twice in a transaction containing a JSON Duality View, then only one logical change will exist, containing an `INSERT` with the updated record.
- **JSON Duality Views to JSON Duality Views Replication using Underlying Tables:** Replication changes at the underlying relational table level of Duality Views are mostly efficient. A Duality View that is on top of the underlying relational table automatically reflects those changes as it is a view. In this case, no additional supplemental logging on the Duality Views is needed.

For steps to set up table-level supplemental logging, see [Enable Table-level Supplemental Logging](#).

Remove Table-level Supplemental Logging

If a table is no longer required to be captured by Oracle GoldenGate and the `TABLE` parameter for the table has been removed from the Extract parameter file, or `TABLEEXCLUDE` is used to exclude the table from a wildcard statement, then supplemental logging can be removed from the table.

Note

If the Extract resolves a table that does not have supplemental logging enabled, it will abend depending on the type of DML operation.

Using `DELETE TRANDATA` to remove supplemental logging sets the Replicat Identity level of the table to `NOTHING`. Supplemental logging can be disabled using the Microservices Architecture web interface from the Administration Service, Configuration page, under the Credential created for a source database, or can be issued with the `DELETE TRANDATA` command.

The following is the syntax for issuing `DELETE TRANDATA`.

```
DBLOGIN USERIDALIAS alias_name
DELETE TRANDATA schema.tablename
```

To check the level of supplemental logging:

```
INFO TRANDATA schema.tablename
```

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 AI Database Product Family supported with the current Oracle GoldenGate release.

Oracle AI 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

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 imply 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`).

From Oracle GoldenGate 21c onward, DML on tables that are not supported will be automatically skipped when `DBA_GOLDENGATE_SUPPORT_MODE.SUPPORT_MODE= NONE` is set. However, DDLs for these objects are still captured based on the `DDL INCLUDE/EXCLUDE` settings. See [Details of Support for Objects and Operations in Oracle DDL](#) for DDL support.

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 Extract, with `NOUSERID` you 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. 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.

Detailed support information for Oracle data types, objects, and operations starts with the following:

Extract Redo Support:

The following data types allow capturing directly from the redo logs and do not require any fetching. If used in a downstream mining configuration, the NOUSERID parameter may be used.

- NUMBER, BINARY FLOAT, BINARY DOUBLE, and (logical) UROWID
- DATE and TIMESTAMP
- VECTOR (a new data type used by AI)
- CHAR, VARCHAR2, LONG, NCHAR, NVARCHAR2, BOOLEAN
- 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
- Native JSON datatype identified by the DTYJSON code.
- Oracle GoldenGate 23ai and higher support piece-wise JSON updates called JSON diff.
- 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, SDO_GEORASTER, or ST_GEOMETRY 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

Data Types Fetched from the Database

Data types listed here are not readable in the redo logs and must be fetched by the Extract process during its processing. The method for fetching these records is controlled by the use of the FETCHOPTIONS parameter.

It is recommended that the database that is generating the redo data is the same database that Oracle GoldenGate uses to fetch the data. However, if this is not possible, an Active Data Guard Standby database open for read-only can also be used as the fetch database.

SECUREFILE LOBs

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

Object tables contain the following attributes:

- Nested table
- SDO_TOPO_GEOMETRY

- SDO_GEORASTER

Fetch does not support ANYDATA columns in a UDT.

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
 - * Part of a key or unique index
 - * In a KEYCOLS clause of the TABLE or MAP parameter
 - * Resolution columns in a CDR (conflict detection and resolution)
 - If an extended VARCHAR column is part of unique index or constraint, then direct path inserts to this table may cause Replicat to abend with a warning. Verify that the extended VARCHAR caused the abend 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;`
- 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.

- 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, 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.

- For JSON datatype, `DTYJSON` is stored in the binary JSON format for query and space efficiency as well as transportability between platforms. A column with JSON data as text is declared using any of the text data types (`VARCHAR2`, `CLOB`) and the `IS JSON` constraint. JSON datatype is supported by Oracle GoldenGate Extract, and Replicat processes along with XStream Out, XStream In processes. JSON support limits the inline text JSON to 4K to prevent Replicat from abending.

By default Extract writes native JSON columns in text format but using `binary_json_format` parameter forces to write in native format. So, this parameter must not be set for `VARCHAR2`, `NVARCHAR2`, `CLOB`, `NCLOB`. The parameter is not set by default. If you are only replicating from Oracle to Oracle you can set the parameter and gain a bit of performance. Also the Column manipulation functions like `str` are supported only for text JSON.

- It is recommended that de-duplication is removed for LOB data types on the target database. If `DEDUPLICATION` is left enabled, it causes severe performance impact on the apply side.

SQLEXEC Limitations

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:

Datatypes Supported By SQLEXEC	Support Limitations
NUMBER, BINARY FLOAT, BINARY DOUBLE UROWID	Special cases of: • XML types • UDTs • Object tables • Collections or nested tables Supports Infinity/-Infinity special values. For more information on data types by databases that support special values, refer to Special Values Supported by Database Data Types . Supports NaN special value except the NUMBER data type.
(N)CHAR, (N) VARCHAR2 LONG, RAW, LONG RAW (N)CLOB, CLOB, BLOB, SECUREFILE, BASICFILE and BFILE	Not supported

Datatypes Supported By SQLEXEC	Support Limitations
XML columns, XMLType	Not supported
Native JSON datatype	VARCHAR2, NVARCHAR2, CLOB, NCLOB not supported with the Extract parameter <code>binary_json_format</code> .
UDT	Not supported
ANYDATA	Not supported
Hierarchy-enabled tables	Not supported
RET Types	Not supported
DICOM	Not supported
SDO_TOPO_GEOMETRY, SDO_GEORASTER	Not supported
Identity columns	Not supported
SDO_RDF_TRIPLE_S	Not supported

Note

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

Note

Any datatype not listed in the table is fully supported by SQLEXEC with the same limitations as the regular product.

Handling Special Data Types

Here are the special configuration requirements for different Oracle data types for Extract.

Support for Lock-free Reservation

Starting with Oracle AI Database 26ai, Oracle GoldenGate supports lock-free reservation. Replication is available for the following combinations between source and target tables:

- Source tables using lockfree reservation columns to the target tables *with* implicit delta conflict resolution for replication from lock-free reservable columns on the source to the reservable columns on the target.
- Source tables without lock-free reservable columns to the target tables with lock-free reservable columns with the following restriction:
 - Primary Key Updates on the source tables without lock-free reservable columns are NOT supported if the target table contains lock-free reservable columns.
- Source tables and target tables using lock-free reservable columns with implicit delta conflict resolution for replication from lock-free reservable columns.

Note

- ACDR is not supported.
- Any manual CDR that results to an UPDATE operation at the target is not supported. For example, UPDATE conflict, INSERTROWEXISTS is not supported.

Support for Data Guard PDB (DGPDB)

Oracle Database 21c introduced the Data Guard PDB (DGPDB) feature, to expand the Data Guard functionality to individual pluggable database (PDB) level. Individual PDBs from a multitenant source database (source CDB) can be configured for Data Guard protection in a target multitenant database (target CDB).

Starting with Oracle GoldenGate 23ai for Oracle AI Database, a PDB can be the source for DGPDB and Oracle GoldenGate per-PDB Extract, simultaneously. When the corresponding DGPDB goes through a role transition to become the new primary, Oracle GoldenGate per-PDB Extract seamlessly migrates to the new source PDB.

During DGPDB switchover or failover, the per-PDB Extract goes through the following transition:

- Detects the transition: Oracle GoldenGate per-PDB Extract internally identifies its change from the old PDB to the new PDB.
- Redirects Service: The auto-restart capability of Oracle GoldenGate Extract automatically connects to the new source PDB in the target CDB.
- Transitions to New Primary: Oracle GoldenGate finishes processing the old archive (redo) log files available from the old source PDB, establishes a sync point, and starts processing the new archive (redo) log files from the new source PDB.

JSON

Oracle GoldenGate supports the following operations using JSON data type:

- Replication support for full JSON document from Oracle database 21c and higher.
- Differential changes of the JSON object are replicated from Oracle AI Database 26ai and higher.

To know about enabling replication of differential changes for JSON objects, see `ADD TRANDATA`. Also see, `CREATE TABLE logical_replication_clause` with the `PARTIAL JSON` option.

Considerations

- Compatible with when both source and target database versions are Oracle AI Database 26ai .
- Table must have a Primary Key.

Note

For details of support for JSON Duality Views and JSON Collection Tables, see [Details of Support for JSON-Relational Duality Views and JSON Collection Tables](#) .

Multibyte Character Types

Multi-byte characters are supported as part of a supported character set. If the semantics setting of an Oracle source database is BYTE and the setting of an Oracle target is CHAR, use the Replicat parameter SOURCEDEFS in your configuration, and place a definitions file that is generated by the DEFGEN utility on the target. These steps are required to support the difference in semantics, whether or not the source and target data definitions are identical. Replicat refers to the definitions file to determine the upper size limit for fixed-size character columns.

TIMESTAMP

To replicate timestamp data, Oracle Database normalizes `TIMESTAMP WITH LOCAL TIME ZONE` data to the local time zone of the database that receives it, the target database in case of Oracle GoldenGate. To preserve the original time stamp of the data that it applies, Replicat sets its session to the time zone of the source database. You can override this default and supply a different time zone by using the `SOURCETIMEZONE` parameter in the Replicat parameter file. To force Replicat to set its session to the target time zone, use the `PRESERVETARGETTIMEZONE` parameter.

To prevent Oracle GoldenGate from abending on `TIMESTAMP WITH TIME ZONE` as TZR, use the Extract parameter `TRANLOGOPTIONS` with `INCLUDEREGIONIDWITHOFFSET` to replicate `TIMESTAMP WITH TIMEZONE` as TZR from an Oracle source that is at least version 10g to an earlier Oracle target, or from an Oracle source to a non-Oracle target. This option allows replicating to Oracle versions that do not support `TIMESTAMP WITH TIME ZONE` as TZR and to database systems that only support time zone as a UTC offset.

You can also use the `SOURCETIMEZONE` parameter to specify the source time zone for data that is captured by an Extract that is earlier than version 12.1.2. Those versions do not write the source time zone to the trail.

Large Objects (LOB)

The following are some configuration guidelines for Extract LOBs.

1. Store large objects out of row if possible.
2. Extract captures LOBs from the redo log. For UPDATE operations on a LOB document, only the changed portion of the LOB is logged. To force whole LOB documents to be written to the trail when only the changed portion is logged, use the `TRANLOGOPTIONS` parameter with the `FETCHPARTIALLOB` option in the Extract parameter file. When Extract receives partial LOB content from the logmining server, it fetches the full LOB image instead of processing the partial LOB. Use this option when replicating to a non-Oracle target or in other conditions where the full LOB image is required.

XML

The following are tools for working with XML within Oracle GoldenGate constraints.

- Although Extract does not support the capture of changes made to an XML schema, you may be able to evolve the schemas and then resume replication of them without the need for a resynchronization, see [Supporting Changes to XML Schemas](#).
- Extract captures XML from the redo log. For UPDATE operations on an XML document, only the changed portion of the XML is logged if it is stored as `OBJECT RELATIONAL` or `BINARY`. To force whole XML documents to be written to the trail when only the changed portion is logged, use the `TRANLOGOPTIONS` parameter with the `FETCHPARTIALXML` option in the Extract parameter file. When Extract receives partial XML content from the logmining server, it fetches the full XML document instead of processing the partial XML. Use this option when replicating to a non-Oracle target or in other conditions where the full XML image is required.

Supporting Changes to XML Schemas

Learn about supporting changes to an XML schema. Extract does not support the capture of changes made to an XML schema.

Supporting RegisterSchema

RegisterSchema can be handled by registering the schema definition on both source and target databases before any table is created that references the XML schema.

Supporting DeleteSchema

Issue DeleteSchema on the source database first.

After Replicat is caught up with the changes made to the source database, issue the DeleteSchema call on the target database.

Supporting CopyEvolve

The CopyEvolve procedure evolves, or changes, a schema and can modify tables by adding or removing columns.

The CopyEvolve procedure can also be used to change whether or not XML documents are valid. Handling CopyEvolve requires more coordination.

Use the following procedure if you are issuing CopyEvolve on the source database.

1. Quiesce changes to dependent tables on the source database.
2. Execute the CopyEvolve on the primary or source database.
3. Wait for Replicat to finish applying all of the data from those tables to the target database.
4. Stop Replicat.
5. Apply the CopyEvolve on the target database.
6. Restart Replicat.

User Defined Types

If Oracle Database is compatible with releases greater than or equal to 12.0.0.0.0, then Extract captures data from redo (no fetch), see [#unique_433](#).

If replicating source data that contains user-defined types with the NCHAR, NVARCHAR2, or NCLOB attribute to an Oracle target, use the HAVEUDTWITHNCHAR parameter in the Replicat parameter file. When this type of data is encountered in the trail, HAVEUDTWITHNCHAR causes Replicat to connect to the Oracle target in AL32UTF8, which is required when a user-defined data type contains one of those attributes. HAVEUDTWITHNCHAR is required even if NLS_LANG is set to AL32UTF8 on the target. By default Replicat ignores NLS_LANG and connects to an Oracle Database in the native character set of the database. Replicat uses the OCIStrring object of the Oracle Call Interface, which does not support NCHAR, NVARCHAR2, or NCLOB attributes, so Replicat must bind them as CHAR. Connecting to the target in AL32UTF8 prevents data loss in this situation. HAVEUDTWITHNCHAR must appear before the USERID or USERIDALIAS parameter in the parameter file.

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 indexTable 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
- Oracle GoldenGate does not support replication of identity column data or Valid Time Temporal column data.

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

Details of Support for JSON-Relational Duality Views and JSON Collection Tables

Oracle GoldenGate 23ai (23.6) supports various replication use cases for JSON Relational Duality Views and JSON Collection Tables. The following list focusses on these scenarios:

- **COMPATIBLE paramter for JSON-Relation Duality Views and JSON Collection Tables:** The source and target Oracle Database must have the COMPATIBLE parameter set to 23.4.0.0.0 or higher in order enable support for JSON Relational Duality Views and JSON Collection Tables. This feature also requires Oracle GoldenGate 23.6 or higher
- **JSON-Relational Duality Views and/or JSON Collection Tables to Data Streams:** Oracle GoldenGate can replicate changes made to JSON Duality View as JSON objects to GoldenGate Data Streams.
- **JSON Collection Tables to JSON Collection Tables:** Oracle GoldenGate can replicate data between JSON Collection Tables.
- **Duality Views to Collection Tables:** Oracle GoldenGate can replicate data from Duality Views to Collection Tables, essentially moving data between these two JSON-focused features.
- **Duality Views to Duality Views Replication between Underlying Tables:** Oracle GoldenGate allows replicating the underlying tables from Duality Views to Duality Views, without replicating the view. If supplemental logging is enabled for a JSON Duality Views, it generates extra redo on top of the existing relational table.

Note

Replication from one Duality View to another is not supported.

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_436](#).

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 Blockchain and Immutable Tables

Starting with Oracle AI Database 26ai, immutable and blockchain tables are supported. Here are the supported operations for the feature:

Support for One-Way Replication

- Both blockchain table versions v1 (introduced in Oracle Database 19.10) and v2 (introduced in Oracle AI Database 26ai) as well as both immutable table versions v1 (introduced in Oracle Database 19.11) and v2 (introduced in Oracle AI Database 26ai) are supported for all Replicats except of coordinated Replicat.
- Interoperability between types and versions:
 - Support for replication of blockchain and immutable tables of any version at the source to conventional tables at the target.
 - No Support for replication of conventional tables at the source to any blockchain and immutable tables of any version at the target.
- DML, DDL, and Procedural Replication must be enabled. For Procedure Replication, the Blockchain table must be appended.
- Delete operations of expired rows are not replicated. Expiration is managed by each database system separately.

Support for Bidirectional (Active-Active) Replication

- Support for version v2 only.
- Support for parallel Replicat in integrated mode and integrated Replicat only.
- ACDR must be configured. ACDR for blockchain and immutable tables contains the following resolution methods:
 - Latest Timestamp
 - Earliest Timestamp

Note

No other resolution methods (such as Column group or delta resolution) is supported. There is no Support for manual CDR.

- A blockchain or immutable table can have at most one primary key or unique constraint and cannot have unique indexes.

- Primary key or unique key constraint on blockchain or immutable tables for ACDR uses non-unique indexes.
- ACDR only manages INSERT-INSERT conflicts because blockchain and immutable tables are INSERT-only tables. ACDR adds two hidden columns for timestamp (CDRTS\$ROW) and visibility (ORABCTAB_INVISIBLE\$).

These hidden columns work as follows:

ACDR will resolve a conflict and retain both copies of the row. It will mark a conflicting INSERT that would have normally been discarded or overwritten as **logically invalid** by setting the hidden column ORABCTAB_INVISIBLE\$ to **1**. As a result, there may be multiple rows with the same key value, but at most 1 row for each key will be **valid** and have ORABCTAB_INVISIBLE\$ set to 0.

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.

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

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 [PLUGGABLE] 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.

Oracle Autonomous AI Database

Prepare your database for Oracle GoldenGate, including configuring connections and logging, enabling Oracle GoldenGate in your database, setting up the flashback query, and managing server resources.

Oracle GoldenGate can be used to replicate data into Oracle Autonomous AI Database from any certified source platform and replicate data from Oracle Autonomous AI Database to any certified target platform.

Different Scenarios for Using Oracle GoldenGate with Oracle Autonomous AI Databases

Using Oracle GoldenGate in the Oracle Autonomous AI 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 AI Database systems.
- **Big Data Integration:** With Oracle GoldenGate for Big Data you can replicate data from the Oracle Autonomous AI 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 AI Database instance back to on-premise or to another cloud database or platform.

See Always Free Autonomous AI Database for details.

Prerequisites for Capturing from an Autonomous AI Database

Oracle Autonomous AI Database has a tight integration with Oracle GoldenGate. There are a number of differences when setting up Extract for an Autonomous AI Database instance compared to a traditional Oracle Database.

Oracle Autonomous AI Database security has been enhanced to ensure that Extract is only able to capture changes from the specific tenant it connected to. Therefore, Downstream Integrated Extract is not supported.

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 and already has all the necessary permissions for both Extract and Replicat processes. 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.

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.

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;
```

Enable Minimal Supplemental Logging in Oracle Autonomous AI Database for Oracle GoldenGate Extract

- Oracle Autonomous AI Database environment is provisioned and running.
- Autonomous AI 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 Oracle Autonomous AI 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 AI 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 AI 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 Oracle Autonomous AI Database instance.

Next Steps After Setting Up Extract Prerequisites for Autonomous AI Database

After preparing Oracle GoldenGate for Oracle Autonomous AI Databases, you can begin follow the instructions in the topic [Configure Extract for Oracle Autonomous Database](#).

Prerequisites for Applying to Oracle Autonomous Database

You can replicate into the Oracle Autonomous AI Database from any source database or platform that is certified by Oracle GoldenGate.

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 Oracle Autonomous AI Database target.
- Oracle Autonomous AI Database is environment provisioned and running.

To deliver data to the Oracle Autonomous AI Database instance using Oracle GoldenGate, perform the following tasks:

Unlock the Autonomous AI Database User for Oracle GoldenGate

Here are the steps to complete the configuration tasks:

Note

Instructions are based on the assumption that the source environment is already configured. Learn the steps required to configure replication into the Oracle Autonomous AI Database environment.

1. For Oracle GoldenGate on-premises, make sure that Oracle GoldenGate is installed.
You can also use Oracle GoldenGate Microservices Architecture 21c for Marketplace for Oracle Autonomous AI Database Serverless 21c. Oracle GoldenGate Microservices Architecture 21c and higher support Autonomous Database capture using Marketplace for Oracle Autonomous AI Database Serverless.
2. Create a deployment for your Oracle GoldenGate environment. This is the deployment where the Replicat that applies data into the Autonomous AI Database will be created.
3. The Autonomous Database instance has a pre-created user for Oracle GoldenGate on-premise called ggadmin. The ggadmin user has been granted the required privileges for Replicat. 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 your Oracle Autonomous AI Database instance as the ADMIN user using any SQL client tool. See About Connecting to Autonomous Database Instance.
4. Run the ALTER USER command to unlock the ggadmin user and set the password for it. This will be used in the command line for any DBLOGIN operations on the Oracle Autonomous AI Database. It will be used in Replicat to allow Oracle GoldenGate to connect to the Oracle Autonomous AI Database and apply data.

```
ALTER USER ggadmin IDENTIFIED BY p0$$word ACCOUNT UNLOCK;
```

Obtain the Autonomous Database Client Credentials

To establish a connection with an Oracle Autonomous AI 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 AI 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 AI 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 AI Database instance.

Next Steps After Setting Up Replicat Prerequisites for Autonomous AI Database

After preparing Oracle GoldenGate for Oracle Autonomous AI Databases, you can begin by following the instructions in the topic [Configure Replicat for Oracle Autonomous AI Database](#).

Details of Support When Using Oracle GoldenGate with Autonomous Databases

Review the supported data types and limitations before replicating data to or from an Oracle Autonomous AI Database instance.

Oracle GoldenGate is supported for any type of Oracle Autonomous AI Database.

Details of Support for coexistence of Oracle GoldenGate with Transient Logical Rolling Upgrades

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 AI Databases

These are the limitations of Oracle GoldenGate when replicating to or from the Oracle Autonomous AI Database.

Supported Replicats

To replicate data into an Oracle Autonomous AI Database you must use parallel Replicat (integrated or non-integrated mode), Classic Replicat or Coordinated Replicat. Integrated Replicat is not supported with Oracle Autonomous AI Database.

Data Type Limitations for DDL and DML Replication

See the section [Non-Supported Oracle Data Types](#).

Also see Data Types in the *Using Oracle Autonomous AI Database Serverless* guide.

DDL replication is supported depending on the restrictions in the Oracle Autonomous AI Database.

Details of Support for Archived Log Retention

The two types of Autonomous Databases, Autonomous AI Database on Dedicated Exadata Infrastructure and Autonomous AI Database Serverless have different log retention behavior.

- Autonomous AI 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.

- **Autonomous AI Database on Dedicated Exadata Infrastructure:** When Oracle Autonomous AI Database 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.

PostgreSQL

Oracle GoldenGate for PostgreSQL supports capture and delivery of initial load and transactional data for supported PostgreSQL database versions.

Oracle GoldenGate for PostgreSQL supports the mapping, filtering, and transformation of source data, unless noted otherwise in this document, as well as replicating data derived from other source databases supported by Oracle GoldenGate, into PostgreSQL databases.

Prepare Database Users and Privileges for PostgreSQL

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

Oracle GoldenGate processes require a database user to capture and deliver data to a PostgreSQL database and it is recommended to create a dedicated PostgreSQL database user for Extract and Replicat.

The following database user privileges are required for Oracle GoldenGate to capture from and apply to a PostgreSQL database.

Privilege	Extract	Replicat	Purpose
Database Replication Privileges			
CONNECT	Yes	Yes	Required for database connectivity. GRANT CONNECT ON DATABASE dbname TO gguser;
WITH REPLICATION	Yes	NA	Required for the user to register Extract with a replication slot. ALTER USER gguser WITH REPLICATION;

Privilege	Extract	Replicat	Purpose
WITH SUPERUSER	Yes	NA	Required to enable table level supplemental logging (ADD TRANDATA) but can be revoked after TRANDATA is enabled for the table(s). ALTER USER gguser WITH SUPERUSER; For Azure Database for PostgreSQL, only the Admin user has SUPERUSER authority and is the only user that can enable TRANDATA.
USAGE ON SCHEMA	Yes	Yes	For metadata access to tables in the schema to be replicated. GRANT USAGE ON SCHEMA tableschema TO gguser;
SELECT ON TABLES	Yes	Yes	Grant select access on tables to be replicated. GRANT SELECT ON ALL TABLES IN SCHEMA tableschema TO gguser;
INSERT, UPDATE, DELETE, TRUNCATE on target tables. Alternatively, if replicating every table, then you can use the GRANT INSERT, UPDATE, DELETE, TRUNCATE ON ALL TABLES IN SCHEMA TO... to the Replicat user, instead of granting INSERT, UPDATE, DELETE to every table.	NA	Yes	Apply replicated DML to target objects. GRANT INSERT, UPDATE, DELETE, TRUNCATE ON TABLE tablename TO gguser;
Heartbeat and Checkpoint Table Privileges			

Privilege	Extract	Replicat	Purpose
CREATE ON DATABASE	Yes	Yes	Required by the Extract and Replicat user to add an Oracle GoldenGate schema for heartbeat and checkpoint table creation. GRANT CREATE ON DATABASE dbname TO gguser; Alternatively, if GGSHEMA is the same as the user, then the objects can be created under the user by issuing CREATE SCHEMA AUTHORIZATION ggsuser;
CREATE, USAGE ON SCHEMA	Yes	Yes	For heartbeat and checkpoint table creation/deletion if the Extract or Replicat user does not own the objects. GRANT CREATE, USAGE ON SCHEMA ggschema TO gguser;
EXECUTE ON ALL FUNCTIONS	Yes	Yes	For heartbeat update and purge function execution if the user calling the functions does not own the objects. GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA ggschema TO gguser;
SELECT, INSERT, UPDATE, DELETE	Yes	Yes	For heartbeat and checkpoint table inserts, updates and deletes if the user does not own the objects. GRANT SELECT, INSERT, UPDATE, DELETE, ON ALL TABLES IN SCHEMA ggschema TO gguser;

To capture data from PostgreSQL standby database specifically, you must have either the SUPERUSER privilege or the pg_read_all_stats privilege, along with the other required privileges to capture data from PostgreSQL database.

Prepare Database Connection

Learn about configuring database connection, system, and parameter settings for Oracle GoldenGate for PostgreSQL.

Oracle GoldenGate for PostgreSQL connects to PostgreSQL databases using a pre-packaged ODBC driver. Connections can be established using a Data Source Name (DSN) or using a direct connection and supplying the database server host, port, database, and other information.

Using DSN connections requires connection details to be listed in an `odbc.ini` file, while using direct entries are entered manually when adding a database connection to the Administration Service's web interface or through the Admin Client.

Note

PgBouncer is not supported for Oracle GoldenGate connections.

Note

Oracle GoldenGate does not support connections to PostgreSQL that use `Pgpool`.

After performing the steps given below to create the DSN entries or you plan to use direct connections, proceed to the [Add Database Connections](#) topic to know how to create database connections.

Configure a DSN Connection in Linux

To create a DSN connection for Oracle GoldenGate processes, you will first need to add an environment variable for the Oracle GoldenGate for PostgreSQL deployment and then create an `odbc.ini` file to store the connection attributes.

1. Log in to the Service Manager web interface.
2. From the left navigation pane, click **Deployments** and then select the Oracle GoldenGate PostgreSQL deployment. This expands the various settings for the deployment.
3. Click **Configuration** and then click the plus sign (+) next to **Environment Variables**.
4. Provide the following information in the two available fields. For the **Environment Variable Value** field, ensure it is the absolute path of the `odbc.ini` file:

Environment Variable Name = `ODBCINI`

Environment Variable Value = `/ogg/deployment/pgsql/odbc.ini`

Add Environment Variable
✕

The Deployment contains the following Environment Variable Settings. Environment Variables can be added, modified, or deleted.

Environment Variable Name	Environment Variable Value
ODBCINI	/ogg/deployment/pgsql/odbc.ini

Submit

5. Click **Submit** to create the new variable and then restart the deployment from the **Deployments** pane, for the changes to take effect.
6. In the Oracle GoldenGate installation's deployment folder, manually create an `odbc.ini` file and add data sources in this file.

Use the following minimum settings when creating the DSN file:

- **Data Source Name** – A user defined name of a source or target database connection that will be referenced by Oracle GoldenGate processes, such as Extract or Replicat. DSN names are allowed up to 32 alpha-numeric characters in length, and can include only underscore (`_`) and dash (`-`) from special characters.
- **IANAAppCodePage=4** – Is the default setting but can be modified according to the guidance specified on the https://docs.progress.com/bundle/datadirect-connect-odbc-71/page/IANAAppCodePage_9.html#IANAAppCodePage_9 page when the database character set is not Unicode.
- **InstallDir** – Is the name of the Oracle GoldenGate PostgreSQL wire protocol driver path, and can use a relative path, like: `InstallDir = ./datadirect`
- **Driver**: Is the name of the Oracle GoldenGate PostgreSQL Wire Protocol driver file, and can use a relative path, like: `Driver= ./datadirect/lib/ggpgsql25.so`.
- **Database**: This is the name of the source or target database.
- **Hostname**: This is the database host IP address or host name.
- **PortNumber**: This is the listening port of the database.
- Oracle GoldenGate for PostgreSQL support multiple hosts and ports for PostgreSQL connections. This feature is useful in the Oracle GoldenGate High Availability (HA) setup. In a PostgreSQL HA setup there is one primary and multiple standby servers. You can set up the connection string format to specify multiple hosts:

Connection String	Description
<code>db-user@host1,host2,host3/db-name</code>	The connections would be attempted using default PostgreSQL database port 5432 for all the hosts (host1, host2, host3).

Connection String	Description
<i>db-user@host1:1234,host2:2345,host3:3456/db-name</i>	The connections would be attempted using the ports associated with each host specified in the connection string.
<i>db-user@host1,host2,host3:1234/db-name</i>	All hosts using the same port number (1234) on all the hosts (host1, host2, host3). The connections would be attempted using the port 1234 for all the hosts (host1, host2, host3).
<i>db-user@host1,host2,host3:1234/db-name</i>	All hosts using the same port number (1234) on all the hosts (host1, host2, host3). The connections would be attempted using the port 1234 for all the hosts.
<i>db-user@host1:3456,host2,host3:1234/db-name</i>	If the connection string is specified in other formats where the port number is specified for some hosts while the other hosts are specified without any ports, then the last or right-most port specification will be used as the port number for all the hosts which do not have a port number specified in the connection string. As shown in the connection string: <i>db-user@host1:3456,host2,host3:1234/db-name</i> The port number for <i>host2</i> would be 1234, which is the right-most port number specified in the connection string.

- You can also provide a LogonID and Password for the Extract or Replicat user, but these will be stored in clear text. It is recommended to leave these fields out of the DSN and instead store them in the Oracle GoldenGate wallet as a credential alias, and reference them with the USERIDALIAS parameter in Extract and Replicat.

The following is an example `odbc.ini` file with two DSN entries. The Data Source names used in the example below are `PG_src` and `PG_tgt`.

```
#[ODBC Data Sources]
```

```
[ODBC]
```

```
IANAAppCodePage=4
```

```
InstallDir=/datadirect
```

```
[PG_src]
```

```
Driver= ./datadirect/lib/ggpsql25.so
```

```
Description=Oracle GoldenGate PostgreSQL Wire Protocol
```

```
Database=sourcedb
```

HostName=remotehost

PortNumber=5432

[PG_tgt]

Driver=../datadirect/lib/ggpgsql25.so

Description=Oracle GoldenGate PostgreSQL Wire Protocol

Database=targetdb

HostName=remotehost

PortNumber=5432

1. Save and close the odbc.ini file.
2. To set up the database connection from Oracle GoldenGate for a PostgreSQL deployment, see [Add Database Connections](#).

Connecting to a FIPS-enabled PostgreSQL System with Version 14 or Lower

When the Oracle GoldenGate Extract is run from a Federal Information Processing Standards (FIPS) enabled system installed with PostgreSQL database lower than version 14, it generates the following error:

```
ERROR OGG-25359 Could not connect to server with database 'postgres', host
'localhost', port '5432' and user name 'postgres'. Error Message: connection to
server at "localhost" (::1), port 5432 failed: could not encrypt password:
disabled for FIPSfe_sendauth: error sending password authentication.
```

The encryption algorithm md5 is the default encryption algorithm on PostgreSQL database version lower than 14 and causes the Extract to abend with an error.

To run Extract on a FIPS-enabled system running PostgreSQL database version lower than 14, perform the following steps:

1. Modify the postgresql.conf file to set the password_encryption option to scram-sha-256.
2. Modify the pg_hba.conf file to set the Method option to scram-sha-256, as md5 is not supported on a FIPS-enabled system. However, this is an optional step.

The password for the database user that is used by Oracle GoldenGate Extract, must be re-generated or modified if the database user has already been created, after the password_encryption option is set to scram-sha-256. You can use the same password to be regenerated.

For example, if the database user, named admin uses the password as password123, then the same password can be regenerated using the scram-sha-256 encryption.

Configuring SSL Support for PostgreSQL

SSL can be enabled by setting the configuration parameter SSL to on in the PostgreSQL configuration file (\$PGDATA/postgresql.conf). If SSL is enabled, the corresponding hostssl entry must be present or added in the pg_hba.conf file.

When SSL is enabled, Oracle GoldenGate uses the root certificate, root certification revocation list (CRL), server client certificate, and key from the default locations, as shown in the following snippet:

```
~/postgresql/root.crt  
~/postgresql/root.crl  
~/postgresql/postgresql.crt  
~/postgresql/postgresql.key
```

You need to create the desired entities from this list, and store them in appropriate locations.

If the SSL configuration is setup using non-default locations, then the following environment variables should be set up as per the environment.

```
PGSSLROOTCERT  
PGSSLCRL  
PGSSLCERT  
PGSSLKEY
```

Changes required in odbc.ini file

The SSL support can be enabled by setting the EncryptionMethod DSN attribute to 1 or 6 in the \$ODBCINI file.

If set to 0 (No Encryption), data is not encrypted.

If set to 1 (SSL), data is encrypted using the SSL protocols specified in the Crypto Protocol Version connection option. If the specified encryption method is not supported by the database server, the connection fails and the driver returns an error.

If set to 6 (RequestSSL), the login request and data are encrypted using SSL if the server is configured for SSL. If the server is not configured for SSL, an unencrypted connection is established. The SSL protocol used is determined by the setting of the Crypto Protocol Version connection option.

If the database server/client certificates also need to be validated, then the corresponding KeyStore file needs to be created and the following ODBC DSN attributes should be setup accordingly in the odbc.ini file.

```
KeyStore=path to .p12 keystore file KeyStorePassword=keystore-passwd  
TrustStore=path to root certificate ValidateServerCertificate=1
```

However, the KeyStore file is not mandatory for validation of client certificates (by the server) when SSLCert and SSLKey are already provided.

Note

Azure Database for PostgreSQL defaults to enforce SSL connections. To adhere to this requirement, perform the requirements listed here, or optionally, you can disable enforcing SSL connections from the Connection security settings of the database instance using the Microsoft Azure Portal.

Database Configuration

Database Software for Capture

To capture from a PostgreSQL database, Oracle GoldenGate requires the `test_decoding` database plug-in be installed for the database. This plug-in might not have been installed by default when the database was installed.

Ensure that the `postgresqlversion#-contrib` package is installed on the database server, as shown in the example:

```
sudo yum install postgresql14-contrib
```

Parameters in the PostgreSQL Database Configuration File

For Oracle GoldenGate, configure the following parameters in the PostgreSQL database configuration file, `$PGDATA/postgresql.conf`:

Note

These changes are required for the primary PostgreSQL database that Oracle GoldenGate will be capturing from as well as any standby database that Oracle GoldenGate could capture from in the event of the primary database failing.

- For remote connectivity of an Extract or Replicat, set the PostgreSQL `listen_addresses` to allow for remote database connectivity. For example:

```
listen_addresses=remotehost_ip_address
```

Note

Ensure that client authentication is set to allow connections from an Oracle GoldenGate host by configuring the `pg_hba.conf` file. For more information, refer to this document: [The pg_hba.conf File](#)

- To support Oracle GoldenGate Extract, write-ahead logging must be set to `logical`, which adds information necessary to support transactional record decoding.

The number of maximum replication slots must be set to accommodate one open slot per Extract, and in general, no more than one Extract is needed per database. If for example PostgreSQL Native Replication is already in use and is using all of the currently configured replication slots, increase the value to allow for the registration of an Extract.

Maximum write-ahead senders should be set to match the maximum replication slots value.

Optionally, commit timestamps can be enabled in the write-ahead log, which when set at the same time logical write-ahead logging is enabled, will track the first DML commit record from that point on, with the correct timestamp value. Otherwise, the first record encountered by Oracle GoldenGate capture will have an incorrect commit timestamp.

```
wal_level = logical                # set to logical for Capture  
  
max_replication_slots = 1          # max number of replication
```

```

slots,
                                # one slot per Extract/client

max_wal_senders = 1            # one sender per max repl slot

track_commit_timestamp = on    # optional, correlates tx commit
time                           # with begin tx log record

(useful for                    # timestamp-based positioning)

```

- After making any of the preceding changes, restart the database.

Use these instructions to manage the database settings for Azure for PostgreSQL, Amazon Aurora PostgreSQL, Amazon RDS for PostgreSQL, Google AlloyDB for PostgreSQL, and Google Cloud SQL for PostgreSQL.

Azure Database for PostgreSQL

When configuring Oracle GoldenGate for PostgreSQL Extract against an Azure Database for PostgreSQL, logical decoding must be enabled and set to LOGICAL.

Read the Microsoft documentation for the instructions:

<https://learn.microsoft.com/en-us/azure/postgresql/>

Other database settings for Azure Database for PostgreSQL can be managed through the Server parameters section of the database instance.

For connections to an Azure Database for PostgreSQL instance, the default Azure Connection Security settings require SSL connections. To adhere to this requirement, further steps are required to support SSL connections with Oracle GoldenGate.

Follow the content listed under [Configuring SSL Support for PostgreSQL](#) for more information.

Amazon Aurora PostgreSQL and Amazon RDS for PostgreSQL

For Amazon Aurora PostgreSQL and Amazon RDS for PostgreSQL, database settings are modified within parameter groups.

Review the Amazon AWS documentation for information on how to edit database settings within a new parameter group and assign it to a database instance:

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_WorkingWithParamGroups.html

- Ensure that the database configuration settings listed previously are correct, by verifying them in the parameter group assigned to the instance.
- The `wal_level` setting for Amazon database services is configured with a parameter called `rds.logical_replication`, whose default is 0 and should be set to 1 if the database is to be used as a source database for Oracle GoldenGate Extract.
- Amazon RDS does not support prepending or starting the replication slot name with the string `rds`. The command used by Oracle GoldenGate Extract to create the replication slot returns an error if the replication slot name starts with the string `rds`. For example, you

could name the replication slot as `ordscdc` instead of `rdscdc` to avoid this error, as shown in the following example:

```
select slot_name from pg_create_logical_replication_slot('ordscdc',  
'test_decoding');
```

This command adds the `ordscdc` replication slot. However, if you use the replication slot name as `rdscdc`, the following error will occur:

```
ERROR: must be superuser or replication role to use logical replication  
slots in the 'rds'  
namespace
```

Limitation:

On Amazon Aurora PostgreSQL version 12.17, if upper case `SHOW` command is executed, it reports the following error:

```
"ERROR: must be superuser or replication role to run this operation."
```

You must use lower case `SHOW` command to avoid this error.

Google AlloyDB for PostgreSQL

Starting with Oracle GoldenGate release 21.14, Oracle GoldenGate supports Google AlloyDB for PostgreSQL. When configuring an Oracle GoldenGate for PostgreSQL Extract for a Google AlloyDB for PostgreSQL, the `alloydb.logical_decoding` configuration parameter (flag) needs to be set to `ON`.

Google Cloud SQL for PostgreSQL

When configuring an Oracle GoldenGate for PostgreSQL Extract for a Google Cloud SQL for PostgreSQL database, logical decoding must be set and is done by setting the `cloudsql.logical_decoding` variable to `ON`. Follow the instructions provided by Google on how to enable this database flag. For more information, see <https://cloud.google.com/sql/docs/postgres/flags#postgres-l>.

Enabling Table-Level Supplemental Logging

Enabling Supplemental logging is a process in which Oracle GoldenGate sets source database table level logging to support change data capture of source DML operations, and depending on the level of logging, to include additional, unchanged columns which would be needed in cases such as bi-directional replication with conflict detection and resolution configured.

There are four levels of table level logging in PostgreSQL, which equate to the `REPLICA IDENTITY` setting of a table, and those include `NOTHING`, `USING INDEX`, `DEFAULT`, and `FULL`.

Oracle GoldenGate requires `FULL` logging for use cases that require uncompressed trail records and Conflict Detection and Resolution, but in cases where tables have a Primary Key or Unique Index whose changes are being replicated in a simple uni-directional configuration or where full before-images or uncompressed records are not needed, then the `DEFAULT` level is acceptable. `NOTHING` and `USING INDEX` logging levels are not supported by Oracle GoldenGate and cannot be set with `ADD TRANDATA`.

The following is the syntax for issuing ADD TRANDATA from the Admin Client.

```
DBLOGIN SOURCEDB dsn_name USERIDALIAS alias_name  
ADD TRANDATA schema.tablename ALLCOLS
```

Note

For tables that have a primary key or unique index, the ALLCOLS option is required in order to set FULL logging for the table, otherwise DEFAULT logging is set.

FULL logging is always set for tables without a primary key or unique index, regardless of whether ALLCOLS is specified or not.

To check the level of supplemental logging:

```
INFO TRANDATA schema.tablename
```

PostgreSQL: Supported Data Types, Objects, and Operations

Oracle GoldenGate for PostgreSQL supports capture and delivery of initial load and transactional data for supported PostgreSQL database versions.

Oracle GoldenGate for PostgreSQL supports the mapping, filtering, and transformation of source data, unless noted otherwise in this document, as well as replicating data derived from other source databases supported by Oracle GoldenGate, into PostgreSQL databases.

Supported Databases

The following are supported databases and limitations for Oracle GoldenGate for PostgreSQL:

- Only user databases are supported for capture and delivery.
- Oracle GoldenGate does not support capture from archived logs.
- Delivery is not supported against replica, standby databases.
- Capture is also supported from replica, standby databases.
- High Availability. See [High Availability Considerations](#) for details.

High Availability Considerations

Oracle GoldenGate supports capturing change data from PostgreSQL High Availability configuration. This functionality is available only on PostgreSQL database versions 16 and higher. The change data capture can be configured on the primary server or any read-only standby server in the High Availability setup. To avoid possible data-loss and manual intervention in cases of failover, Oracle GoldenGate recommends that the High Availability setup should be configured using synchronous replication.

Refer to Oracle GoldenGate procedures for PostgreSQL High Availability Failover Doc ID [2818379.1](#) for more details on possible data loss and manual intervention scenarios.

To configure the CDC Extract from a PostgreSQL High Availability setup, perform the following tasks:

1. Extract can be configured to capture data from the primary server or a read-only standby server in the High Availability setup.
2. When the Extract is configured on one node (primary or standby), the replication slot with the same name should be created on all the other nodes in the High Availability setup at the same time, before starting the data capture.
 - a. Use REGISTER EXTRACT command to create the replication slot on one server (where the Extract is configured), as shown in the following example:

```
REGISTER EXTRACT exte
```

- b. Immediately after registering Extract, create the replication slot with the same name. You must use the same replication slot name on all other nodes in the High Availability setup explicitly.

```
postgres=# select
pg_create_logical_replication_slot('exte_ac6b520cab2871b7',
'test_decoding');
pg_create_logical_replication_slot
-----
(exte_ac6b520cab2871b7,0/8169E00)
(1      row)
```

```
2025-01-20T06:58:22Z INFO      OGG-25355 Successfully created
replication slot 'exte_ac6b520cab2871b7' for Extract group 'EXTE' in
database 'postgres'.
```

- c. Connect to the corresponding database and execute the pg_create_logical_replication_slot statement from PSQL.

Note

IMPORTANT NOTE FOR REPLICATION SLOT CREATION ON STANDBY:

When the replication slot creation is attempted on the read-only standby server (either using REGISTER EXTRACT command or pg_create_logical_replication_slot statement), the operation completes only after some DML (can be any random DML) is executed on any table on the primary server in the high availability setup. Alternatively the heartbeat functionality can be enabled on the primary server, which can cause the required DML activity on the primary, to complete the replication slot creation on the standby server.

3. ADD TRANDATA or DELETE TRANDATA commands are not supported on any read-only standby server(s) in the PostgreSQL high availability setup. The REPLICA IDENTITY setting of any table on the read-only standby server would be the same as the REPLICA IDENTITY setting of the corresponding table on the primary server.
4. To use the heartbeat functionality when the data capture is configured on a read-only standby server, the heartbeat functionality should be enabled on the primary server.
5. In the case of failover, the extract parameter file should be modified to connect to the desired node (either primary/standby, based on the requirement) to continue the data capture after the failover.

Example

Consider a situation, where there are three nodes in the PostgreSQL high availability cluster, with corresponding connection aliases:

Node1 (current primary) with corresponding connection alias: node1

Node2 (standby) with corresponding connection alias: node2

Node3 (standby) with corresponding connection alias: node3

Before the failover, if the Extract is configured to capture data from one of the nodes (such as node2), then the Extract parameter file would contain an entry similar to the following:

```
USERIDALIAS node2
```

In case of a failover, to the primary node (node1) goes down and the secondary node (node2) becomes the new primary then there can be two possibilities:

- a. To continue data capture from same node2:
 - No changes are required in the Extract parameter file.
 - Restart the Extract.
- b. To capture the data always from a different node:
 - Change the Extract parameter file to connect to the desired node, such as Node3.

```
USERIDALIAS node3
```

- Restart the Extract.

Supported PostgreSQL Data Types

Here's a list of PostgreSQL data types that Oracle GoldenGate supports along with the limitations of this support.

- array
- bigint
- bigserial
- bit(n)
- bit varying(n)
- boolean
- bytea
- char (n)
- cidr
- citext
- date
- decimal
- double precision
- Enumerated Types
- inet
- integer

- interval
- json
- jsonb
- macaddr
- macaddr8
- money
- numeric
- pgvector extension
- real
- serial
- smallint
- smallserial
- text
- time with/without timezone
- timestamp with/without timezone
- tsquery
- tsvector
- uuid
- varchar(n)
- varbit
- xml

Handling Array Data Type

PostgreSQL supports array of various built-in and UDT types. Starting with Oracle GoldenGate release 23ai, Oracle GoldenGate Extract and Replicat support PostgreSQL array data type of following types:

- boolean
- bigint
- bit
- char
- date
- double precision
- enum
- int
- interval
- numeric
- money
- real

- smallint
- timestamp, timestampz, time, timetz
- varbit
- varchar
- text
- tsvector
- tsquery

The real, double, numeric data types support the NaN and Infinity/-Infinity special numeric values.

For more information on data types by databases that support special values, refer to [Special Values Supported by Database Data Types](#).

Both initial load and CDC Extract support these array types.

Note

Initial load Extract does not support PostgreSQL array of time datatype.

Limitations of Support

- If columns of char, varchar, text, or bytea data types are part of a primary or unique key, then the maximum individual lengths for these columns must not exceed 8191 bytes.
- Extract cannot process records with bytea columns of more than 512 MB in size.
- Columns of data type CITEXT that are part of the Primary Key are supported up to 8000 bytes in size. CITEXT columns that are greater than 8000 bytes and are part of the Primary Key are not supported.
- Extract cannot process records with bytea columns of more than 512 MB in size.
- decimal: NaN input values are not supported.
- The following limitations apply to bit/varbit data types:
 - They are supported up to 4k in length. For lengths greater than 4k the data is truncated and only the lower 4k bits are captured.
 - The source bit(n) column can be applied only onto a character type column on a non-PostgreSQL target and can be applied onto a char type or a bit/varbit column on PostgreSQL target.
- The following limitations are applicable to both timestamp with time zone and timestamp without time zone:
 - The timestamp data with BC or AD tags in the data is not supported.
 - The timestamp data older than 1883-11-18 12:00:00 is not supported.
 - The timestamp data with more than 4 digits in the YEAR component is not supported.
 - Infinity/-Infinity input strings for timestamp columns are not supported.
- The following are the limitations when using interval:

- The capture of mixed sign interval data from interval type columns is not supported. You can use `DBOPTIONS ALLOWNONSTANDARDINTERVALDATA` in the Extract parameter file to capture the mixed sign interval data (or any other format of interval data, which is not supported by Oracle GoldenGate) as a string (not as standard interval data).

The following are a few examples of data that gets written to the trail file, on using the `DBOPTIONS ALLOWNONSTANDARDINTERVALDATA` in the Extract param file:

- `+1026-9 +0 +0:0:22.000000` is interpreted as 1026 years, 9 months, 0 days, 0 hours, 0 minutes, 22 seconds.
- `-0-0 -0 -8` is interpreted as 0 years, 0 months, 0 days, -8 hours.
- `+1-3 +0 +3:20` is interpreted as 1 year, 3 months, 0 days, 3 hours, 20 minutes.
- Replicat: If the source interval data was captured using `DBOPTIONS ALLOWNONSTANDARDINTERVALDATA` and written as a string to the trail, the corresponding source column is allowed to be mapped to either a char or a binary type column on the target.
- date limitations are:
 - The date data with BC or AD tags in the data is not supported.
 - Infinity/-Infinity input strings for date columns are not supported.
- Columns of text, json, xml, bytea, char (>8191), varchar (>8191) are treated as LOB columns and have the following limitations:
 - When using `GETUPDATEBEFORES`, the before image of LOB columns is never logged.
 - When using `NOCOMPRESSUPDATES`, LOB columns are logged in the after image only if they were modified.
- The support of 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-Supported PostgreSQL Data Types

Oracle GoldenGate for PostgreSQL does not support the following data types:

- box
- bpchar
- circle
- Composite Types
- Domain Types
- line
- lseq
- Object Identifiers Types
- path
- pg_lsn
- pg_snapshot
- point

- polygon
- Pseudo-Types
- Range Types
- User-defined Types (UDTs)
- Extensions and Additional Supplied Modules listed at: <https://www.postgresql.org/docs/current/contrib.html> are not supported by Oracle GoldenGate unless explicitly listed under [Supported PostgreSQL Data Types](#).

Note

If the Extract parameter file contains a table with unsupported data types, the Extract will stop with an error message. To resume replication, remove the table from the Extract file or remove the column from the table with an unsupported data type.

Supported Objects and Operations for PostgreSQL

- Oracle GoldenGate for PostgreSQL only supports DML operations (Insert/Update/Deletes). DDL replication is not supported.
- Oracle GoldenGate for PostgreSQL supports replication of truncate operations beginning with PostgreSQL 11 and above, and requires the GETTRUNCATES parameter in Extract and Replicat.
- Multiple truncate operations are supported in pgoutput plugin only.
- Case-Sensitive/Insensitive names Usage:
 - Unquoted names are case-insensitive and are implicitly lowercase. For example, `CREATE TABLE MixedCaseTable` and `SELECT * FROM mixedcasetable` are equivalent.
 - Quoted table and column names are case-sensitive and need to be listed correctly in Extracts and Replicats and with Oracle GoldenGate commands.

For example, `TABLE appschema."MixedCaseTable"` and `ADD TRANDATA appschema."MixedCaseTable"` would be required to support a case-sensitive table name.

Tables, Views, and Materialized Views

Tables to be included for capture and delivery must meet the following requirements and must only include data types listed under *Supported PostgreSQL Data Types*.

- Oracle GoldenGate for PostgreSQL supports capture of transactional DML from user tables, and delivery to user tables.
- Oracle GoldenGate for PostgreSQL, versions 21.14.0.0.0 and after, supports capture and delivery to base partitioned tables. To capture data from PostgreSQL partitioned tables, the Replica Identity setting of all the partitions *must match* with the Replica Identity setting of the base partitioned table. This is applicable for multi-level partitioned tables as well.

You can choose one of the following ways, to set up this configuration:

1. Run `ADD TRANDATA` command on all or the required partitions of the base-partitioned table:

```
ADD TRANDATA partition-name ALLCOLS
```

2. Run the ALTER TABLE command on all or required partitions of the base-partitioned table:

```
ALTER TABLE partition-name REPLICA IDENTITY replica-identity-setting
```

- Globalization is supported for object names (table /schema/database/column names) and column data.
- Capture and delivery for a materialized view is supported.
- Extract process supports materialized views, which must have unique index and it must be refreshed concurrently.
- The Replica Identity of materialized views must be set to FULL so that CDR, UNIFIED FORMAT RECORD and NOCOMPRESSDELETES, GETUPDATEBEFORES can be supported. Also, the replica identity FULL setting is required for the materialized view to process the WAL DELETE record. If the Replica Identity for the materialized view is not set to FULL then the DELETE record will not be processed by Extract.

To set the Replica Identity to FULL, use the following command:

```
add trandata schema.materialized_view, ALLCOLS
```

- The statistics for an UPDATE or PK UPDATE will log as DELETE followed by INSERT. Although this displays an anomaly in the Extract statistics, but the data sync is correct when the source and target materialized views data is compared.
- The PK UPDATE results in DELETE followed by INSERT so the Extract statistics will show differently than the database row count. You have to rely on the data comparison tools to check the data integrity between source and target data, containing the materialized views records.
- The data from the trail cannot be applied to materialized view by the Replicat process. You can map the materialized views data from the trail to any other table instead.

Limitations

- Oracle GoldenGate for PostgreSQL does not support capture and delivery for views.
- Oracle GoldenGate for PostgreSQL does not support capture from individual partitions of a partitioned table.

Sequences and Identity Columns

- Sequences are supported on source and target tables for unidirectional, bidirectional, and multi-directional implementations.
- Identity columns created using the GENERATED BY DEFAULT AS IDENTITY clause, are supported on source and target tables, for unidirectional, bidirectional, and multi-directional implementations.
- Identity columns created using the GENERATED ALWAYS AS IDENTITY clause, are not supported in target database tables and the Identity property should be removed from target tables or changed to GENERATED BY DEFAULT AS IDENTITY.
- For bidirectional and multi-directional implementations, define the Identity columns and sequences with an INCREMENT BY value equal to the number of servers in the configuration, with a different MINVALUE for each one.

For example, MINVALUE /INCREMENT BY values for a bidirectional, two-database configuration would be as follows:

Database1, set the MINVALUE at 1 with an INCREMENT BY of 2.

Database2, set the MINVALUE at 2 with an INCREMENT BY of 2.

For example, MINVALUE /INCREMENT BY values for a multi-directional, three-database configuration would be as follows:

Database1, set the MINVALUE at 1 with an INCREMENT BY of 3.

Database2, set the MINVALUE at 2 with an INCREMENT BY of 3.

Database3, set the MINVALUE at 3 with an INCREMENT BY of 3.

SQL Server

With Oracle GoldenGate for SQL Server, you can perform initial loads and capture transactional data from supported SQL Server versions and replicate the data to a SQL Server database or other supported Oracle GoldenGate targets, such as an Oracle Database.

Oracle GoldenGate for SQL Server supports data filtering, mapping, and transformations unless noted otherwise in this documentation.

SQL Server Supported Versions

Oracle GoldenGate Extract supports Enterprise Edition and some versions of SQL Server Standard Edition. Review the Exceptions and Additional Information column of the certification matrix to see the details of which Standard Edition versions of SQL Server are supported for capture.

Oracle GoldenGate Replicat supports both SQL Server Enterprise and Standard editions.

Oracle GoldenGate supports remote capture and delivery for Azure SQL Database Managed Instance and remote delivery for Azure SQL Database.

Oracle GoldenGate supports remote capture and delivery for Amazon RDS for SQL Server and Google Cloud SQL for SQL Server.

Prepare Database Users and Privileges for SQL Server

The following database users and privileges are required for Oracle GoldenGate to capture from and apply to a SQL Server database.

Database User for SQL Server

Oracle GoldenGate processes require a database user in order to capture from and apply data to a SQL Server database and it is recommended to create a dedicated database user to be used exclusively by Oracle GoldenGate processes.

Oracle GoldenGate for SQL Server supports SQL Server authentication for all of its certified platforms .

To use SQL Server authentication, create a dedicated SQL Server login for Extract and Replicat and assign the privileges listed below.

Required Database Privileges

SQL Server and Azure SQL Managed Instance

The following user requirements and minimum database privileges and permissions are required for Oracle GoldenGate to capture from and apply to a SQL Server or Azure SQL Managed Instance database.

1. Create a dedicated login for Oracle GoldenGate for SQL Server or Azure SQL Managed Instance.
2. Add the login as a user to the msdb database and to the source or target database.
3. Create a schema in the source or target database, to be used for objects required for Oracle GoldenGate. This schema should map to the GGSHEMA value used in the GLOBALS parameter file.
4. Enable the following privileges and permissions for the Oracle GoldenGate user based on whether the user is for an Extract, or for a Replicat.

Table 7-3 Privileges and Permissions for Oracle GoldenGate User

Privilege	Extract	Replicat	Syntax
msdb Database Roles and Privileges			
SQLAgentReaderRole	Yes	No	ALTER ROLE SQLAgentReaderRole ADD MEMBER <i>gguser</i> ;
SQLAgentUserRole	Inherited	Yes	ALTER ROLE SQLAgentUserRole ADD MEMBER <i>gguser</i> ;
User Database Roles and Privileges			
SYSADMIN	Yes	No	Required for a one time change to enable database level Change Data Capture (CDC) if not already enabled and can be revoked once TRANDATA has been enabled. ALTER SERVER ROLE sysadmin ADD MEMBER <i>gguser</i> ; Database Administrators with sysadmin credentials can manually enable the database for CDC using the following, which would negate the need for the Extract user to have this privilege: EXEC sys.sp_cdc_enable_db;
DBOWNER	Yes	Yes	ALTER ROLE db_owner ADD MEMBER <i>gguser</i> ;

Amazon RDS for SQL Server

The following user requirements and minimum database privileges and permissions are required for Oracle GoldenGate to capture from and apply to an Amazon RDS for SQL Server database:

1. Create a dedicated login for Oracle GoldenGate for Amazon RDS for SQL Server.
2. Add the login as a user to the msdb database and to the source or target database.
3. Create a schema in the source or target database, to be used for objects required for Oracle GoldenGate. This schema should map to the GGSHEMA value used in the GLOBALS parameter file.
4. Enable the following privileges and permissions for the Oracle GoldenGate user based on whether the user is for an Extract, or for a Replicat.

Table 7-4 Privileges and Permissions for Oracle GoldenGate User

Privilege	Extract	Replicat	Syntax
msdb Database Roles and Privileges			
EXECUTE ON rds_cdc_enable_db	Yes	No	GRANT EXECUTE ON msdb.dbo.rds_cdc_enable_db TO <i>gguser</i> ; Database administrators with master credentials can manually enable the database for Change Data Capture using the following command, which would negate the need for the Extract user to have this permission: EXEC msdb.dbo.rds_cdc_enable_db 'source_database'
SQLAgentOperatorRole	Yes	No	ALTER ROLE SQLAgentOperatorRole ADD MEMBER <i>gguser</i> ;
SQLAgentUserRole	Inherited	Yes	ALTER ROLE SQLAgentUserRole ADD MEMBER <i>gguser</i> ;
User Database Roles and Privileges			
DBOWNER	Yes	Yes	ALTER ROLE db_owner ADD MEMBER <i>gguser</i> ;

Azure SQL Database

The following user requirements and minimum database privileges and permissions are required for Oracle GoldenGate to apply to an Azure SQL Database:

1. Create a dedicated login for Oracle GoldenGate for Azure SQL Database.
2. Add the login as a user to the target database.
3. Create a schema in the target database, to be used for objects required for Oracle GoldenGate. This schema should map to the GGSHEMA value used in the GLOBALS parameter file.
4. Enable the following privileges and permissions for the Oracle GoldenGate user.

Table 7-5 Privileges and Permissions for Oracle GoldenGate User

Privilege	Extract	Replicat	Syntax
User Database Roles and Privileges			
DBOWNER	NA	Yes	ALTER ROLE db_owner ADD MEMBER <i>gguser</i> ;

Google Cloud SQL for SQL Server

The following user requirements and minimum database privileges and permissions are required for Oracle GoldenGate to capture from and apply to a Google Cloud SQL for SQL Server database:

1. Create a dedicated login for Oracle GoldenGate Google Cloud SQL for SQL Server. The user must be created from within the **Users** section of the Google Cloud dashboard for the database instance.

2. Add the user to the source or target database.
3. Create a schema in the source or target database, to be used for objects required for Oracle GoldenGate. This schema should map to the GGSHEMA value used in the GLOBALS parameter file.
4. If the database is to be used as a source for an Extract, manually enable the database for Change Data Capture (CDC):


```
EXEC msdb.dbo.gccloudsql_cdc_enable_db 'source_database';
```
5. Enable the following privileges and permissions for the Oracle GoldenGate user based on whether the user is for an Extract, or for a Replicat.

Table 7-6 Privileges and Permissions for Oracle GoldenGate User

Privilege	Extract	Replicat	Syntax
User Database Roles and Privileges			
DBOWNER	Yes	Yes	ALTER ROLE db_owner ADD MEMBER gguser;

Prepare Database Connection

Learn about configuring database connection, system, and parameter settings for Oracle GoldenGate for SQL Server.

Oracle GoldenGate for SQL Server connects to SQL Server databases using a pre-packaged ODBC driver. Connections can be established using a Data Source Name (DSN) or using a direct connection and supplying the database server host, port, database, and other information.

Using DSN connections requires connection details to be listed in an `odbc.ini` file, while using direct entries are entered manually when adding a database connection to the Administration Service's web interface or through the Admin Client.

Configure a DSN Connection in Linux

The following steps demonstrate how to create a DSN connection on Linux using an `odbc.ini` file.

1. For Oracle GoldenGate 23ai versions prior to 23.6, you will first need to add a new ODBCINI environment variable for the SQL Server deployment using the steps below. Versions beginning with 23.6 automatically set this variable during deployment creation.
 - a. Log in to the Service Manager web interface.
 - b. From the left navigation pane, click **Deployments** and then select the Oracle GoldenGate SQL Server deployment. This expands the various settings for the deployment.
 - c. Click **Configuration** and then click the plus sign (+) next to **Environment Variables**.
 - d. Provide the following information in the two available fields. For the **Environment Variable Value** field, ensure it is the absolute path of the `odbc.ini` file:


```
Environment Variable Name = ODBCINI
Environment Variable Value = ${OGG_HOME}/msodbc/odbc.ini
```
 - e. Click **Submit** to create the new variable and then restart the deployment from the **Deployments** pane for the changes to take effect.

2. In the Oracle GoldenGate installation's msodbc folder, manually create an `odbc.ini` file and add data sources in this file using the following information and example.

Example for Oracle GoldenGate 23ai versions prior to 23.6, where Driver must be set to 'ODBC Driver 18 for SQL Server'

```
[mssql_source]
Driver = ODBC Driver 18 for SQL Server
Server = myserver,1433
Database = source_database
TrustServerCertificate=yes
```

Example for Oracle GoldenGate 23ai versions 23.6 or later, where Driver must be set to 'Microsoft ODBC Driver for SQL Server'

```
[mssql_source]
Driver = Microsoft ODBC Driver for SQL Server
Server = myserver,1433
Database = source_database
TrustServerCertificate=yes
```

Note

For a complete list of available connection options, review the Microsoft documentation at the following link:
<https://learn.microsoft.com/en-us/sql/connect/odbc/dsn-connection-string-attribute?view=sql-server-ver16>

3. Save and close the `odbc.ini` file.
4. To create the database connection for the deployment, see [Add Database Connections](#).

Configure a DSN Connection in Windows

Before creating a database connection for Oracle GoldenGate processes running on Windows, install the latest version of either Microsoft ODBC Driver 17 for SQL Server or the Microsoft ODBC Driver 18 for SQL Server.

Follow these steps to create a system DSN on the Windows server where Oracle GoldenGate is installed.

To create a SQL Server DSN:

1. Open the ODBC Data Sources (64-bit) application.
2. In the ODBC Data Source Administrator dialog box, select the System DSN tab, and then click **Add**.
3. Under Create New Data Source, select the ODBC Driver {version} for SQL Server and then click **Finish**. The Create a New Data Source to SQL Server wizard appears.
4. Enter the following details, and click **Next**:
 - Name: Can be of your choosing. In a Windows cluster, use one name across all nodes in the cluster.

- Description: (Optional) Type a description of this data source.
 - Server: Type the SQL Server connection string or server\instance name. For Always On connections, use the listener\instance name of the Always On Availability Group.
5. For login authentication, select **SQL Server authentication**, specify the **Login ID** and **Password** information, and then click **Next**.
 6. If configuring a connection using the Microsoft ODBC Driver 18 for SQL Server, click **Next** again to go to the last configuration page. Select the option for **Trust server certificate** and then click **Back** to proceed as you need to first enable the Trust server certificate before selecting the default database in the next step.
 7. Select **Change the default database to**, and then select the source or target database from the list. Enable the **Use ANSI** settings, and click **Next**.
 8. Leave the next page of the wizard as-is and click **Finish**.
 9. Click **Test Data Source** to test the connection.
 10. If the test is successful, close the confirmation box and the Create a New Data Source box.
 11. Repeat this procedure for each SQL Server source and target database, where Oracle GoldenGate process will connect.

Connecting to the Listener of a SQL Server Always On Configuration

Extract and Replicat can connect to the listener of an Always On configuration or directly to the current primary replica of the group.

The advantage of creating the connection to the listener is that Extract or Replicat can reconnect to the new primary replica upon failover without having to reconfigure the connection string to the new primary.

An Extract can also be configured to route its read-only queries to an available readable, synchronous mode secondary replica. By default, if Extract connects to a listener, all processing will be done against the primary replica, but if an Extract is configured with the `TRANLOGOPTIONS ALWAYSONREADONLYROUTING` parameter, its read-only queries are routed by the listener to an available readable secondary replica.

See `TRANLOGOPTIONS` and [Requirements Summary for Capture and Delivery of Databases in an Always On Availability Group](#) for more information.

If creating the DSN to connect to a Listener of an Always On configuration, enable the Multi-subnet failover option when creating a DSN. For Linux DSN connections, use the `MultiSubnetFailover=Yes` option in the DSN entry.

Preparing Tables for Processing

The table attributes in the following sections must be addressed in your Oracle GoldenGate environment.

Replicat Consideration for Target Identity Columns, Triggers, and Constraints

When replicating data to a target SQL Server database that has identity columns, triggers, and cascade and check constraints, consider the following:

- For columns that have an identity column, Replicat sets the `IDENTITY_INSERT ON` for the table, which may reduce delivery performance.

- For tables that have triggers or cascade constraints, execution of the trigger or cascade operation may result in a Replicat error if the Replicat is configured to deliver the same data that a trigger will insert or cascade constraint will update or delete.

For example, TableA on the source has a trigger that inserts a record into TableB. The Extract is configured to capture records for both TableA and TableB. On the target, the Replicat will first insert a record for TableA, then the trigger for TableA fires and inserts into TableB, followed by the Replicat attempting to insert the same record into TableB, which will result in a Replicat error.

- Check any foreign key constraints are also enforced, which may reduce delivery performance.
- For tables on the target database that have triggers, set the SET XACT_ABORT parameter to off. This ensures that execution of the trigger operation does not result in missing transactions.

To overcome these situations, there are several options that can be implemented based on the replication use case.

- For unidirectional implementations where a Replicat is the only process writing data to the target tables, consider the following options for Identity columns, triggers and constraints on the target tables.
 - Disable or drop the Identity property, triggers and constraints on the target tables.
 - Modify the identity property, triggers and constraints and set the NOT FOR REPLICATION option on for each and ensure that the Microsoft ODBC driver is at least version 17.8.1.
- For multi-directional implementations where both a Replicat and application write data to the target tables, and triggers and constraints are enabled, modify the Identity property, triggers and constraints and set the NOT FOR REPLICATION option on for each and ensure that the Microsoft ODBC driver is at least version 17.8.1.

Additionally, to use IDENTITY columns in a multi-directional replication configuration, define the IDENTITY columns to have an increment value equal to the number of servers in the configuration, with a different seed value for each one.

For example, a three-database configuration would be as follows:

Database1 set the seed value at 0 with an increment of 3.

Database2 set the seed value at 1 with an increment of 3.

Database3 set the seed value at 2 with an increment of 3.

Improving IDENTITY Replication with Array Processing

Because only one table per session can have IDENTITY_INSERT set to ON, Replicat must continuously toggle IDENTITY_INSERT when it applies IDENTITY data to multiple tables in a session. To improve the performance of Replicat in this situation, use the BATCHSQL parameter. BATCHSQL causes Replicat to use array processing instead of applying SQL statements one at a time.

Ensuring Row Uniqueness in Source and Target Table

Oracle GoldenGate requires some form of 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:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. 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 those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration.

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.

4. If a table does not have an appropriate key, or if you prefer that the existing key(s) are not 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. See TABLE | MAP in *Reference for Oracle GoldenGate*.

Using KEYCOLS to Specify a Custom Key

If a table does not have an applicable row identifier, or if you prefer that identifiers are not used, you can define a substitute key, providing that 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 overrides any existing primary or unique key that Oracle GoldenGate finds.

Prepare the Database for Oracle GoldenGate

Learn how to enable supplemental logging in the source database tables that are to be used for capture by the Extract for SQL Server and how to purge older CDC staging data.

You can learn more about CDC Capture using the following:

Using the Oracle GoldenGate for SQL Server CDC Capture Replication http://www.oracle.com/webfolder/technetwork/tutorials/obe/fmw/goldengate/12c/sql_cdcprep/sql_cdcprep.html.

Enabling CDC Supplemental Logging

With the CDC Extract, the method of capturing change data is via SQL Server Change Data Capture tables, so it is imperative that you follow the procedures and requirements below, so that change data is correctly enabled, maintained, and captured by Extract.

You will enable supplemental logging with the ADD TRANDATA command so that Extract can capture the information that is required to reconstruct transactions.

ADD TRANDATA must be issued for all tables that are to be captured by Oracle GoldenGate, and to do so requires that a valid schema be used in order to create the necessary Oracle GoldenGate tables and stored procedures.

Enabling supplemental logging for a CDC Extract does the following:

- Enables SQL Server Change Data Capture at the database level, if it's not already enabled.
- Creates a Change Data Capture staging table for each base table enabled with supplemental logging by running `EXECUTE sys.sp_cdc_enable_table`, and creates a trigger for each CDC table. The CDC table exists as part of the system tables within the database and has a naming convention like, `cdc.OracleGG_basetableobjectid_CT`.
- Creates a tracking table of naming convention `ggschema.OracleGGTranTables`. This table is used to store transaction indicators for the CDC tables and is populated when the trigger for a CDC table is fired. The table will be owned by the schema listed in the GLOBALS file's, `GGSCHEMA` parameter.
- Creates a unique fetch stored procedure for each CDC table, as well as several other stored procedures that are required for Extract to function. These stored procedures will be owned by the schema listed in the GLOBALS file's, `GGSCHEMA` parameter.
- Also, as part of enabling CDC for tables, SQL Server creates two jobs per database:
`cdc.dbname_capture`
`cdc.dbname_cleanup`
- The CDC Capture job is the job that reads the SQL Server transaction log and populates the data into the CDC tables, and it is from those CDC tables that the Extract will capture the transactions. So it is of extreme importance that the CDC capture job be running at all times. This too requires that SQL Server Agent be set to run at all times and enabled to run automatically when SQL Server starts.

Note

If SQL Server Transactional Replication is also enabled for the database, the CDC Capture job will not exist and instead, only the SQL Server Log Reader Agent job will exist.

- The CDC Capture job can be tuned for better throughput and tuning information can be found in [CDC Capture Method Operational Considerations](#).
- The CDC Cleanup job that is created by Microsoft does not have any dependencies on whether the Oracle GoldenGate Extract has captured data in the CDC tables or not. Therefore, extra steps are needed to disable or delete the the CDC cleanup job immediately after `TRANDATA` is enabled, and to enable Oracle GoldenGate's own Purge Change Data task. See [Purge CDC Staging Data](#) for more information.

To enable supplemental logging using the command line interface, use the following high-level steps:

1. Review the Prepare Database Users and Privileges topic in order to determine required privileges and steps to enable the database for Change Data Capture, if it is not already set. Elevated permissions may be needed for GoldenGate if the database is not enabled for CDC but can be negated by having an admin manually enable the database for Change Data Capture.
 - For Google Cloud SQL for SQL Server , the database has to manually be enabled for Change Data Capture by a service admin user and executing the following command:
`EXEC msdb.dbo.gccloudsql_cdc_enable_db 'source_database';`

- For SQL Server and Azure SQL Managed Instance, adding TRANDATA will attempt to set the database for Change Data Capture if the user has sysadmin privileges, otherwise a database administrator can manually enable the database for CDC prior to adding TRANDATA, by executing the following command against the source database:

```
EXEC sys.sp_cdc_enable_db;
```
 - For Amazon RDS for SQL Server, adding TRANDATA will also attempt to set the database for Change Data Capture if the user has been granted the permission, otherwise a database administrator with master credentials can manually enable the database for CDC prior to adding TRANDATA, by executing the following command against the source database:

```
EXEC msdb.dbo.rds_cdc_enable_db 'source_database'
```
2. In the source Oracle GoldenGate installation, ensure that the GLOBALS file has the parameter GGSHEMA *schemaname* and that the schema name used has been created (CREATE SCHEMA *schemaname*) in the source database. This schema will be used by all subsequent Oracle GoldenGate components created in the database, therefore it is recommended to create a unique schema that is solely used by Oracle GoldenGate, such as 'ggschema' and to not use the SQL Server schemas dbo or cdc.
 3. On the source Oracle GoldenGate system, open the command line interface (Admin Client).
 4. Connect to the database with the database login credentials.
 5. Issue the following command for each table that is to be captured by an Extract. You can use a wildcard to specify multiple table names.

```
ADD TRANDATA owner.table
```

```
ADD TRANDATA owner.*
```

Optionally, you can designate the filegroup in which the SQL Server Change Data Capture staging tables will be placed, by using the FILEGROUP option with an existing filegroup name.

```
ADD TRANDATA owner.table FILEGROUP cdctablesSee ADD TRANDATA
```

See [ADD TRANDATA](#) for more details.

Purge CDC Staging Data

When enabling supplemental logging, data that is required by Extract to reconstruct transactions are stored in a series of SQL Server CDC system tables, as well Oracle GoldenGate objects that are used to track operation ordering within a transaction. These tables require routine purging in order to reduce data storage within the database. As part of enabling supplemental logging using TRANDATA, SQL Server creates its own Change Data Capture Cleanup job, however this job is unaware that an Extract may still require data from these CDC system tables and can remove that data before the Extract has a chance to capture it.

If data that Extract needs during processing has been deleted from the CDC system tables, then one of the following corrective actions might be required:

- Alter Extract to capture from a later point in time for which CDC data is available (and accept possible data loss on the target).
- Resynchronize the source and target tables, and then start the Oracle GoldenGate environment over again.

To remedy the situation of CDC data being removed before an Extract can process it, Oracle GoldenGate for SQL Server requires that a Purge Change Data task be created. This task will purge CDC staging data while ensuring that no data is purged that the Extract has yet to process.

Use the following steps immediately after enabling supplemental logging (TRANDATA) and prior to starting the Extract, to create the Oracle GoldenGate Purge Change Data task. The Purge Change Data task runs within the Administration Service and will automatically delete the SQL Server CDC Cleanup job when first created. There is no SQL Server Agent job for the Purge Change Data task as it is run by the Administration Service. Therefore, the Administration Service must be running in order for the cleanup task to function correctly.

To create a Purge Change Data task:

With Oracle GoldenGate for Microservices Architecture, after adding TRANDATA to tables but prior to starting Extract, a **Purge Change Data** task must be created to perform CDC cleanup on the database. This can be done through either one of the following:

- Manual REST API requests
- Administration Server Web UI

To create a **Purge Change Data** task using the Administration Service Web UI:

1. Expand the **Tasks** section in the left pane.
2. Click **Purge Change Data** from the **Tasks** page.
3. Click the plus sign to display a form, and fill out the required fields to create a new Purge Change Data task.
 - a. **Operation Name:** Name of the purge task to be created.
 - b. **Enabled:** Set the task to enabled, which is the default value.
 - c. **Credential Domain and Credential Alias:** Select an existing Credential Alias for the source database.
 - d. **Keep Rule:** This value determines in hours or days, the amount of CDC staging data to keep in the source database. Depending on the version of Oracle GoldenGate, the default values are either 3 days or 1 hour. Lower CDC data retention periods reduce the amount of CDC staging data stored in the database but limit the ability for a user to reposition the Extract back to a time older than the data that exists in the staging tables.
 - e. **Purge Frequency:** This value represents how often the task runs, with a default value of every 10 minutes. It is recommended to keep the default value unless overhead from the purge task is impacting database performance during periods of high user activity.

Note

Only create one Purge Change Data task per source database.

Additional information of the Oracle GoldenGate CDC Cleanup job can be found in [CDC Capture Method Operational Considerations](#).

Enabling 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.

To create a Filter Table and enable Supplemental Logging:

The steps below require a database user who is a member of the SQL Server System Administrators (sysadmin) role.

1. On the source system, run Admin Client.
2. Connect to the deployment from the Admin Client.
3. Issue the following command to log into the database.
`DBLOGIN USERIDALIAS alias`
4. Create the Oracle GoldenGate checkpoint table that is used by the Replicat to deliver data to the source database.

Example: `ADD CHECKPOINTTABLE ogg.ggchkpt`

It is recommended that you use the same schema name as used in the `GGSCHEMA` parameter of the `GLOBALS` file.

5. Enable supplemental logging for the newly created checkpoint table.

Example: `ADD TRANDATA ogg.ggchkpt`

6. Add the Replicat with the checkpoint table information.

Example: `ADD REPLICAT repe, EXTTRAIL north/ea,checkpointtable ogg.ggchkpt`

7. Configure the Extract with the `EXCLUDEFILERTABLE` parameter, using the Replicat's checkpoint table for the filtering table.

`TRANLOGOPTIONS EXCLUDEFILERTABLE ogg.ggchkpt`

CDC Capture Method Operational Considerations

Learn about the SQL Server CDC Capture options, features, and recommended settings.

Tuning SQL Server Change Data Capture

The following information is useful in improving the capture performance of the Extract.

- 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.
- Ensure that Auto Create Statistics and Auto Update Statistics are enabled for the database. Maintaining statistics on the `cdc.OracleGG_#####_CT` tables, `cdc.lsn_time_mapping` table, and `OracleGGTranTables` table are crucial to the performance and latency of the Extract.

- The SQL Server Change Data Capture job collects data from the SQL Server transaction log and loads it into the Change Data Capture staging tables within the database.

As part of the job that is created, there are several available tuning parameters that can be used, and information on how to best tune the job can be found in the following article:
[https://technet.microsoft.com/en-us/library/dd266396\(v=sql.100\).aspx](https://technet.microsoft.com/en-us/library/dd266396(v=sql.100).aspx)

As a general recommendation, you should change the SQL Server Change Data Capture Job polling interval from the default of 5 seconds to 1 second.

To change the default polling interval of the CDC Capture job, execute the following queries against the database:

```
EXEC [sys].[sp_cdc_change_job]
@job_type = N'capture',
@pollinginterval = 1,
GO,
--stops cdc job
EXEC [sys].[sp_cdc_stop_job],
@job_type = N'capture',
GO,
--restarts cdc job for new polling interval to take affect
EXEC [sys].[sp_cdc_start_job],
@job_type = N'capture',
```

Oracle GoldenGate CDC Object Versioning

Oracle GoldenGate provides a version tracking subsystem to track the CDC objects that are created by Oracle GoldenGate when enabling supplemental logging. These objects are:

- Oracle GoldenGate change tracking tables in the format `OracleGG_object_id_CT`.
- Stored procedures in the format `fetch_database_name_object_id`
- Stored procedures `OracleCDCExtract`, `OracleGGCreateProcs`, and `OracleGGCreateNextBatch`.
- After successfully completing the `ADD TRANDATA` command, Oracle GoldenGate creates a table called `OracleGGVersion` under the `GGSCHEMA` specified in the `GLOBALS` file, if it does not already exist.

Next, Oracle GoldenGate inserts a record into the table that tracks the start and end time of the `TRANDATA` session. When Extract starts up, it checks for consistency between itself and the Oracle GoldenGate CDC objects by comparing its internal version number with the version numbers found in the `OracleGGVersion` table. If it finds that the version numbers do not match, it abends with a message similar to the following:

```
ERROR OGG-05337 The Oracle GoldenGate CDC object versions on database, source,
are not consistent with the expected version, 6. The following version(s)
were found: 3. Rerun ADD TRANDATA for all tables previously enabled, including
heartbeat, heartbeat seed, and filter tables.
```

Supported and Unsupported Extract Parameters for SQL Server Change Data Capture

Learn about the parameters used with CDC Extract. The following table lists the supported and unsupported parameters.

Supported Parameters	Unsupported Parameters
TRANLOGOPTIONS	TRANLOGOPTIONS MANAGESECONDARYTRUNCATIONPOINT/ NOMANAGESECONDARYTRUNCATIONPOINT/ ACTIVESECONDARYTRUNCATIONPOINT
ALWAYSONREADONLYROUTING	-
TRANLOGOPTIONS LOB_CHUNK_SIZE	-
TRANLOGOPTIONS MANAGECDCCLEANUP/ NOMANAGECDCCLEANUPTRANLOGOPTIONS	-
TRANLOGOPTIONS QUERYTIMEOUT	-
TRANLOGOPTIONS TRANCOUNT	-

TRANLOGOPTIONS LOB_CHUNK_SIZE

The Extract parameter LOB_CHUNK_SIZE is added for the CDC Capture method to support large objects. If you have huge LOB data sizes, then you can adjust the LOB_CHUNK_SIZE from the default of 4000 bytes, to a higher value up to 65535 bytes, so that the fetch size is increased, reducing the trips needed to fetch the entire LOB.

Example: TRANLOGOPTIONS LOB_CHUNK_SIZE 8000

TRANLOGOPTIONS MANAGECDCCLEANUP/NOMANAGECDCCLEANUP

The Extract parameter MANAGECDCCLEANUP/NOMANAGECDCCLEANUP is used by the CDC Capture method to instruct the Extract on whether or not to maintain recovery checkpoint data in the Oracle GoldenGate CDC Cleanup job. The recovery checkpoint is maintained, but purging of the ct table(s) will not happen if the SQL Server CDC cleanup job is also not enabled. It is required for purging.

The default value is MANAGECDCCLEANUP and it doesn't have to be explicitly listed in the Extract. However, it does require creating the Oracle GoldenGate CDC Cleanup job prior to starting the Extract. MANAGECDCCLEANUP should be used for all production environments, where NOMANAGECDCCLEANUP may be used for temporary and testing implementations as needed.

Example: TRANLOGOPTIONS MANAGECDCCLEANUP

TRANLOGOPTIONS EXCLUDEUSER/EXCLUDETRANS

The SQL Server CDC Capture job does not capture user information or transaction names associated with a transaction, and as this information is not logged in the CDC staging tables, Extract has no method of excluding DML from a specific user or DML of a specific transaction name. The EXCLUDEUSER and EXCLUDETRANS parameters are therefore not valid for the CDC Capture process.

**TRANLOGOPTIONS MANAGESECONDARYTRUNCATIONPOINT/NOMANAGESECONDARYTRUNCATIONPOINT/
ACTIVESECONDARYTRUNCATIONPOINT**

The SQL Server Change Data Capture job is the only process that captures data from the transaction log when using the Oracle GoldenGate CDC Capture method. The secondary truncation point management is not handled by the Extract, and for the Change Data Capture Extract, these parameters are not valid.

TRANLOGOPTIONS ALWAYSONREADONLYROUTING

The ALWAYSONREADONLYROUTING parameter allows Extract for SQL Server to route its read-only processing to an available read-intent Secondary when connected to an Always On availability group listener.

TRANLOGOPTIONS QUERYTIMEOUT

Specifies how long queries to SQL Server will wait for results before reporting a timeout error message. This option takes an integer value to represent the number of seconds. The default query timeout value is 300 seconds (5 minutes). The minimum value is 0 seconds (infinite timeout). The maximum is 2147483645 seconds.

TRANLOGOPTIONS TRANCOUNT

Allows adjustment of the number of transactions processed per each call by Extract to pull data from the SQL Server change data capture staging tables. Based on your transaction workload, adjusting this value may improve capture rate throughput, although not all workloads will be positively impacted. The minimum value is 1, maximum is 100, and the default is 10.

Details of the Oracle GoldenGate CDC Cleanup Process

The Oracle GoldenGate Purge Change Data task is required for a CDC Extract by default, since Extract defaults to TRANLOGOPTIONS MANAGECDCCLEANUP.

There should only be one purge task for each database enabled for CDC Capture, and you must create the task using the steps mentioned in the [Prepare the Database for Oracle GoldenGate](#) section of this document.

Modifying the Oracle GoldenGate Purge Change Data Task

The default purge task frequency schedule, and data retention period for the Oracle GoldenGate Purge Change Data task is to run every 10 minutes, with a data retention policy of 3 days or 1 hour, depending on the version of Oracle GoldenGate installed.

For customer specific requirements, it may be necessary to adjust the retention period (**Keep Rule** option) and the task run-time schedule (**Purge Frequency** option).

The **Keep Rule** option determines in hours or days, the amount of CDC staging data to keep in the source database. Depending on the version of Oracle GoldenGate installed, the default values are either 3 days or 1 hour. Lower CDC data retention periods reduce the amount of CDC staging data stored in the database but limit the ability for a user to reposition the Extract back to a time older than the data that exists in the staging tables. Typically, there would be no need to reposition an existing Extract back to an earlier point in time, so it is recommended to use the newer default setting of 1 hour unless there is a specific case that requires more staging data to remain in the database. Note that though if you change this value from a higher retention period to a very short retention period, the next time the task schedule runs, it could consume a lot of transaction log space and system overhead. So it is recommended to slowly decrease the **Keep Rule** value over time, until you reach the desired ending value.

The **Purge Frequency** represents how often the task runs, with a default of every 10 minutes. It is recommended to keep the default value unless overhead from the purge task is impacting database performance during periods of high user activity.

To modify an existing Purge Change Data task, navigate to the **Configuration** from the menu on the left of the Administration Service, to open the **Configuration** page.

1. Click **Tasks** from the **Configuration** page to open the **Tasks** page.

2. Click **Purge Change Data** from the **Tasks** page.
3. Click the **Alter Task** icon next to an existing task.
4. Modify the values of **Keep Rule** and **Purge Frequency** options as required.
5. Click **Submit** to save the changes.

Deleting the Oracle GoldenGate Purge Change Data Task

Deleting a Purge Change Data task for a database should only be done if there are no Extracts configured to capture against the specific database.

To delete an existing Purge Change Data task, navigate to the **Configuration** option from the menu on the left of the Administration Service, to open the **Configuration** page.

1. Click **Tasks** from the **Configuration** page to open the **Tasks** page.
2. Click **Purge Change Data** from the **Tasks** page.
3. Click the **Delete Task** icon next to the task to be removed.

Updating from Classic Extract to a CDC Extract

If you plan to change from using a Classic Extract from Oracle GoldenGate 12c (12.3.0.1) or earlier, to an Oracle GoldenGate 23ai CDC Extract, then you must remove the supplemental logging that was implemented using the Classic Extract installation method, and re-enable supplemental logging using the CDC Extract installation binaries, as the calls to enable TRANDATA are different between the two versions, and the implementation of TRANDATA for Classic Extract is not supported by the CDC Extract.

Follow these general guidelines to remove and re-enable supplemental logging. Special consideration and planning should be involved if migrating from Classic to CDC Extract in a production system. The information provided here does not cover all requirements and is only offered as general requirements regarding supplemental logging:

1. Ensure that the Classic Extract has processed all remaining data in the logs and can be gracefully stopped.
2. Do one of the following, depending on how Extract was running in relation to other replication or CDC components:
 - If Extract was *not* running concurrently with SQL Server transactional replication or a non-Oracle CDC configuration on the same database, open a query session in Management Studio and issue the following statement against the source database to disable and delete any CDC or replication components, and to clear the secondary truncation point.


```
EXEC sys.sp_cdc_disable_db
```
 - If Extract was running *concurrently* with SQL Server transactional replication or a non-Oracle CDC configuration on the same database, run Admin Client from the Classic Extract's installation folder, login to the source database with the DBLOGIN, and then issue the following command for each table that is in the Extract configuration. You can use a wildcard to specify multiple table names


```
DELETE TRANDATA owner.table
DELETE TRANDATA owner.*
```
3. Delete any heartbeat table entries if one was installed.

```
DELETE HEARTBEATTABLE
```

4. Using the Oracle GoldenGate CDC Extract installation binaries, follow the steps listed in [Prepare the Database for Oracle GoldenGate](#) to re-enable supplemental logging and other necessary components, and re-add the heartbeat table.

Requirements Summary for Capture and Delivery of Databases in an Always On Availability Group

Oracle GoldenGate for SQL Server supports capture from a primary replica or a read-only, synchronous mode secondary replica of an Always On Availability Group, and delivery to the primary replica.

When capturing from either a primary or a secondary replica in an Always On Availability Group, it is important to understand that the capture process must only read hardened transactions from the log, and that there be no potential for data loss between any replica database that Oracle GoldenGate is or will capture from.

Database Connection

For both Extract and Replicat, it is recommended to create a database connection that uses the Always On Availability Group Listener for the connection.

- For the Replicat, connecting to the Listener allows the Replicat to reconnect if the primary replica performs a failover to a new instance, without having to manually edit the connection settings to point to the new primary.
- For the Extract connecting to the Listener not only allows reconnecting to the primary without editing the connection to point to the new instance, but also provides the optional ability to run the Extract's data extraction stored procedures, against a read-only secondary.
- For both Extract and Replicat connected to an Always On environment, use the AUTORESTART parameter for the Manager, to restart the processes after a failover.
- To route the Extract's data extraction queries to a read-only secondary, ensure that the connection uses the Listener, that you have one or more read-only secondary replicas that are configured to handle read-only routing, and that the Extract runs with the `TRANLOGOPTIONS ALWAYSONREADONLYROUTING` parameter.

Refer to the following Microsoft documentation on how to configure read-only routing:
<https://docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/configure-read-only-routing-for-an-availability-group-sql-server?view=sql-server-2017>

Supplemental Logging

Supplemental logging must be enabled by normal means (`ADD TRANDATA`) using Admin Client or the web interface connected to the primary replica and not against a secondary replica.

- Create a DSN to the primary replica, or to the Always On Availability Group Listener, to connect using `DBLOGIN` to run `ADD TRANDATA`.
- When enabling supplemental logging against the primary replica database, the SQL Server Change Data Capture job does not automatically get created on any secondary replicas. Upon failover from a primary to a secondary, you must manually create the SQL Server Change Data Capture job and the Oracle CDC Cleanup job if in use, on the new primary replica.

```
EXECUTE sys.sp_cdc_add_job N'capture
```


- When creating the SQL Server CDC Capture job on the new primary, the default configuration settings are put in place. So if you have previously modified the default values on the former primary replica, you need to run `sys.sp_cdc_change_job` on the new primary and set the values accordingly.

Note

Consult the [Microsoft documentation](#) on how to enable the CDC Capture job for AlwaysOn Secondary Replicas for more information.

Operational Requirements and Considerations

- When an instance is no longer the primary instance but has the SQL Server CDC Capture job installed, the job ceases to run after some time and does not attempt to restart. Upon failover back to that instance, the job does not automatically start, so it must be manually started.
- If secondary replica databases are not in sync with the primary replica database, the CDC capture job will not advance in the log, and therefore no records will be captured by an Extract, until such time that the primary and secondary replicas are synchronized. See this article from Microsoft for more details:

<https://docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/replicate-track-change-data-capture-always-on-availability?view=sql-server-2017>

Note

When capturing from either a primary or a secondary replica in an Always On Availability Group, it is important to understand that the capture process must only read hardened transactions from the log, and that there be no potential for data loss between any replica database that Oracle GoldenGate is or will capture from.

- When running an Extract from a middle tier Windows or Linux server, set the middle tier server's date, time, and time zone to the same as that of the primary replica.
- If Extract is configured to capture from a readable secondary replica, but not configured with read-only routing, the SQL Server CDC Capture job must be created against the secondary replica prior to starting the Extract, as the Extract will check if the job exists. To create the SQL Server CDC Capture job, any potential secondary that will have an Extract connected to it, must at some point be set to a writable Primary database and then follow the steps above, under supplemental logging, to manually add the SQL Server CDC Capture job.
- If uninstalling Oracle GoldenGate and disabling Change Data Capture on a database that is part of an Always On availability group, follow the extra steps provided in [Disabling Change Data Capture](#).

SQL Server: Supported Data Types, Objects, and Operations

Learn about support information for Oracle GoldenGate on SQL Server Database.

With Oracle GoldenGate for SQL Server supports capture and delivery of initial load and transactional data for supported SQL Server database versions.

Oracle GoldenGate for SQL Server supports the mapping, filtering, and transformation of source data, unless noted otherwise in this document, as well as replicating data derived from other source databases supported by Oracle GoldenGate, into SQL Server databases.

Instance Requirements

- The SQL Server server name (@@SERVERNAME) must not be NULL.
- (Extract) For Oracle GoldenGate to capture transactional data, the SQL Server Agent must be running on the source SQL Server instance and the SQL Server Change Data Capture job must be running against the database. If SQL Server Transactional Replication is also enabled for the database, then the SQL Server Log Reader Agent must be running.
- If your data for TEXT, NTEXT, IMAGE, or VARCHAR(MAX), NVARCHAR(MAX) and VARBINARY(MAX) columns will exceed the SQL Server default size set for the max_text_repl_size option, then extend the size. Use sp_configure to view or adjust the current value of max_text_repl_size.

Note

For Amazon RDS for SQL Server, to adjust instance settings, you need to use Parameter Groups instead of sp_configure.

- It is recommended to install the most recent Service Pack or Cumulative Update for your SQL Server instance to ensure proper functionality. For SQL Server 2012, 2014, 2016, and 2017, Microsoft has identified and fixed several important issues that directly affect the SQL Server Change Data Capture feature. This situation impacts the ability for Oracle GoldenGate to correctly capture data. The current known issues that require Microsoft patches include KB3030352, KB3166120, and KB4073684.

Database Requirements

Observe the following requirements and limitations for supporting Oracle GoldenGate:

- Only user databases are supported for capture and delivery.
- Ensure that Auto Create Statistics and Auto Update Statistics are enabled for the database.
- The database must be set to the compatibility level of the SQL Server instance version.
- Oracle GoldenGate supports SQL Server databases configured with Transparent Data Encryption (TDE).
- (Extract) The source database can be set to any recovery model that supports the change data capture feature in Microsoft SQL Server.
- If the source database was created by restoring a backup from a different instance you must synchronize the database owner SID with the SID on the new instance. Alternatively, you can use sp_changedbowner to set the restored database to a current login.
- Capture from SQL Server databases (SQL Server 2017 CU15 and higher releases) enabled with In-Memory OLTP (in-memory optimization) is supported, with Oracle GoldenGate releases 21.15 and 23.5 onwards. However, only capture from on-disk tables is supported, and not from the memory optimized tables.
- (AlwaysOn) Extract supports capturing from the primary database, or a read-only, synchronous-commit mode. Asynchronous-commit mode are not supported for capture.

- Replicat performance consideration: Beginning with SQL Server 2016, Microsoft changed the default setting for the database option TARGET_RECOVERY_TIME from 0 to 60 seconds. It has been demonstrated in internal testing that this can reduce the Replicat's throughput. If you experience Replicat throughput degradation, consider adjusting the TARGET_RECOVERY_TIME setting to 0.

Limitations:

- Oracle GoldenGate does not support capture or delivery of system databases.
- Oracle GoldenGate does not support capture from contained databases.
- Source database names cannot exceed 121 characters. This limitation is due to the SQL Server stored procedures that are used to enable supplemental logging.
- If you are configuring the Oracle GoldenGate heartbeat functionality, the SQL Server database name must not exceed 107 characters.
- (AlwaysOn) Capture from databases configured in asynchronous-commit mode of an AlwaysOn Availability group are not supported.

Table Requirements

Tables to be included for capture and delivery must include only the data types that are listed in [Supported SQL Server Data Types](#).

- Oracle GoldenGate supports capture of transactional DML from user tables, and delivery to user tables and writeable views.
- DDL operations are not supported.
- Oracle GoldenGate supports the maximum permitted table names and column lengths for tables that are tracked by SQL Server Change Data Capture.
- The sum of all column lengths for a table to be captured from must not exceed the length that SQL Server allows for enabling Change Data Capture for the table. If the sum of all column lengths exceeds what is allowed by SQL Server procedure `sys.sp.cdc_enable_table`, then ADD TRANDATA cannot be enabled for that table. The maximum allowable record length decreases as more columns are present, so there is an inverse relationship between maximum record length and the number of columns in the table.

Supported SQL Server Data Types

The following data types are supported for capture and delivery, unless specifically noted in the limitations that follow:

- Binary Data Types
 - (binary, varbinary, varbinary (max))
 - (varbinary (max)with FILESTREAM)
- Character Data Types
 - (char, nchar, nvarchar, nvarchar (max), varchar, varchar (max))
- Date and Time Data Types
 - (date, datetime2, datetime, datetimeoffset, smalldatetime, time)
- Numeric Data Types

- (bigint, bit, decimal, float, int, money, numeric, real, smallint, smallmoney, tinyint)
- LOBs
 - (image, ntext, text)
- Other Data Types
 - (timestamp, uniqueidentifier, hierarchyid, geography, geometry, sql_variant (Delivery only), XML)
- Oracle GoldenGate for SQL Server can replicate column data that contains SPARSE settings..

Limitations:

- Oracle GoldenGate does not support filtering, column mapping, or manipulating large objects larger than 4KB. Full Oracle GoldenGate functionality can be used for objects of up to 4KB.
- Oracle GoldenGate treats XML data as a large object (LOB), as does SQL Server when the XML does not fit into a row. SQL Server extended XML enhancements (such as lax validation, DATETIME , union functionality) are not supported.
- A system-assigned TIMESTAMP column or a non-materialized computed column cannot be part of a key. A table containing a TIMESTAMP column must have a key, which can be a primary key or unique constraint, or a substitute key specified with a KEYCOLS clause in the TABLE or MAP statements. For more information see Assigning Row Identifiers.
- Oracle GoldenGate supports multibyte character data types and multi byte data stored in character columns. Multibyte data is supported only in a like-to-like, SQL Server configuration. Transformation, filtering, and other types of manipulation are not supported for multibyte character data.
- If capture of data for TEXT, NTEXT, IMAGE, VARCHAR (MAX), NVARCHAR(MAX) and VARBINARY (MAX) columns will exceed the SQL Server default size set for the max text repl size option, extend the size. Use sp_configure to view the current value of max text repl size and adjust the option as needed.

Note

Amazon RDS for SQL Server does not allow max text repl size to be greater than 64MB.

- Columns of IMAGE, NTEXT, and TEXT data types are logged as a NULL value for delete and before image update operations. Columns of VARBINARY(MAX), VARCHAR(MAX), and NVARCHAR(MAX) are logged as a NULL value for before image update operations unless the column was updated.

For more information, review the Large Object Data Types content in the following Microsoft document:

<https://docs.microsoft.com/en-us/sql/relational-databases/system-tables/cdc-capture-instance-ct-transact-sql?view=sql-server-ver15>

- Oracle GoldenGate supports UDT and UDA data of up to 2 GB in size. All UDTs except SQL_Variant are supported.
- Common Language Runtime (CLR), including SQL Server built-in CLR data types (such as, geometry, geography, and hierarchy ID), are supported. CLR data types are supported

only in a like-to-like SQL Server configuration. Transformation, filtering, and other types of manipulation are not supported for CLR data.

- VARBINARY (MAX) columns with the FILESTREAM attribute are supported up to a size of 4 GB. Extract uses standard Win32 file functions to read the FILESTREAM file.
- The range and precision of floating-point numbers depends on the host machine. In general, 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.
- Oracle GoldenGate supports time stamp data from 0001/01/03:00:00:00 to 9999/12/31:23:59:59. If a time stamp is converted from GMT to local time, these limits also apply to the resulting time stamp. Depending on the time zone, conversion may add or subtract hours, which can cause the time stamp to exceed the lower or upper supported limit.

Limitations on Computed Columns:

- Computed columns, either persisted or non-persisted, are not supported by Microsoft's Change Data Capture. Therefore, no data is written to the trail for columns that contain computed columns. To replicate data for non-persisted computed columns, use the FETCHCOLS or FETCHMODCOLS option of the TABLE parameter to fetch the column data from the table.

Keep in mind that there can be discrepancies caused by differences in data values between the time that the column was changed in the database and the time that Extract fetches the data for the transaction record that is being processed.

- When a column with unsupported data type `sql_variant` is used in the FETCHCOLS or FETCHMODCOLS options, Extract abends with following error:

```
ERROR   OGG-25230 Unsupported datatype 'column-name (sql_variant)' used
with 'FETCHCOLS/FETCHCOLSEXCEPT' in table 'table-name>.
```

- Replicat does not apply DML to any computed column, even if the data for that column is in the trail, because the database does not permit DML on that type of column. Data from a source persisted computed column, or from a fetched non-persisted column, can be applied to a target column that is not a computed column.
- In an initial load, all of the data is selected directly from the source tables, not the transaction log. Therefore, in an initial load, data values for all columns, including non-persisted computed columns, is written to the trail or sent to the target, depending on the method that is used. As when applying change data, however, Replicat does not apply initial load data to computed columns, because the database does not permit DML on that type of column.
- Oracle GoldenGate does not permit a non-persisted computed column to be used in a KEYCOLS clause in a TABLE or MAP statement.
- If a unique key includes a non-persisted computed column and Oracle GoldenGate must use the key, the non-persisted computed column is ignored. This may affect data integrity if the remaining columns do not enforce uniqueness.
- If a unique index is defined on any non-persisted computed columns, it is not used.
- If a unique key or index contains a non-persisted computed column and is the only unique identifier in a table, Oracle GoldenGate must use all of the columns as an identifier to find target rows. Because a non-persisted computed column cannot be used in this identifier, Replicat may apply operations containing this identifier to the wrong target rows.

Non-Supported SQL Server Data Types and Features

- SQL_Variant data type is not supported for capture.
- Tables that contain unsupported data types may cause Extract to Abend. As a workaround, you must remove TRANDATA from those tables and remove them from the Extract's TABLE statement, or use the Extract's TABLEEXCLUDE parameter for the table.

Supported Objects and Operations for SQL Server

The following objects and operations are supported:

- Parallel Replicat is supported with Oracle GoldenGate for SQL Server.
- Oracle GoldenGate supports capture of transactional DML from user tables and delivery to user tables and writeable views.
- TEXT, NTEXT, IMAGE, VARBINARY, VARBINARY (MAX), VARCHAR (MAX), and NVARCHAR (MAX) columns are supported in their full size for operations that are logged by SQL Server Change Data Capture. For example, columns of IMAGE, NTEXT, and TEXT data types are logged as a NULL value for delete operations. For more information, review the Large Object Data Types content at the following Microsoft document:
<https://docs.microsoft.com/en-us/sql/relational-databases/system-tables/cdc-capture-instance-ct-transact-sql?view=sql-server-ver15>
- Oracle GoldenGate supports the maximum row sizes that are permitted for tables that are enabled for SQL Server Change Data Capture.
- Oracle GoldenGate supports capture from tables enabled with PAGE and ROW compression. For partitioned tables that use compression, all partitions must be enabled with the same compression type.
- Oracle GoldenGate supports capture for partitioned tables if the table has the same physical layout across all partitions.
- The sum of all column lengths for a table to be captured from must not exceed the length that SQL Server allows for enabling Change Data Capture for the table. If the sum of all column lengths exceeds what is allowed by the SQL Server procedure `sys.sp.cdc_enable_table`, then ADD TRANDATA cannot be added for that table. The maximum allowable record length decreases as more columns are present, so there is an inverse relationship between maximum record length and the number of columns in the table.

Non-Supported Objects and Operations for SQL Server

The following objects and operations are not supported:

- Consider a parent-child relationship between two tables, where a primary key in the parent table is referenced by a foreign key in the child table with the ON UPDATE SET DEFAULT clause. The deferred PK update on the parent table results in DELETE and INSERT pairs. It is not possible to reliably identify the corresponding DELETE and INSERT operations on the parent table. When such a deferred update is performed on the parent table, Oracle GoldenGate Extract may not reliably capture all events.
- For source databases, operations that are not supported by SQL Server Change Data Capture, such as TRUNCATE statements. Refer to Microsoft SQL Server Documentation for a complete list of the operations that are limited by enabling SQL Server Change Data Capture.

- Oracle GoldenGate for SQL Server does not support the capture or delivery of DDL changes for SQL Server and extra steps are required for Oracle GoldenGate processes on the source and target to handle any table level DDL changes, including table index rebuild operations. See [Requirements for Table Level DDL Changes](#).
- Views are not supported.
- Operations by the TextCopy utility and WRITETEXT and UPDATETEXT statements. These features perform operations that either are not logged by the database or are only partially logged, so they cannot be supported by the Extract process.
- Partitioned tables that have more than one physical layout across partitions.
- Partition switches against a source table. SQL Server Change Data Capture treats partition switches as DDL operations, and the data moved from one partition to another is not logged in the CDC tables, so you must follow the procedures in [Requirements for Table Level DDL Changes](#) to manually implement a partition switch when the table is enabled for supplemental logging.
- Due to a limitation with SQL Server's Change Data Capture, column level collations that are different from the database collation, may cause incorrect data to be written to the CDC tables for character data and Extract will capture them as they are written to the CDC tables. It is recommended that you use NVARCHAR, NCHAR or NTEXT data type for columns containing non-ASCII data or use the same collation for table columns as the database. For more information see, [About Change Data Capture \(SQL Server\)](#).
- Due to a limitation with SQL Server's Change Data Capture, NOOPUPDATES are not captured by the SQL Server Change Data Capture agent so there are no records for Extract to capture for no-op update operations.
- Temporal tables are not supported for enabling Change Data Capture, therefore cannot be configured for Extract for source implementations.

Requirements for Table Level DDL Changes

Oracle GoldenGate for SQL Server does not support the capture or delivery of DDL changes. However, beginning with Oracle GoldenGate 21c, changes made to tables enabled with TRANDATA will not cause Extract to abend. Extract will continue to process change data for the table as it existed when TRANDATA was enabled.

Operations considered to be table-level DDL changes include, but are not limited to: ALTER TABLE, TRUNCATE TABLE, index rebuilds, and partition switches.

To avoid data inconsistencies due to table level DDL changes, the following steps are required.

1. Source: Pause or Stop application data to the table or tables to be modified.
2. Source: Ensure that there are no open transactions against the table to be modified.
3. Source: Ensure that the SQL Server CDC Capture job processes all remaining transactions for the table that is to be modified.
4. Source: Ensure that the Extract processes all the transactions for the table that is to be modified, prior to making any DDL changes.
5. Target: Ensure that the Replicat processes all the transactions for the table that is to be modified, prior to making any DDL changes.
6. Optionally, implementing an Event Marker table can be used to determine when all of the remaining transactions have been processed for the table that is to be modified, and handle the coordination of when to correctly stop the Extract and Replicat.
7. Source: Stop the Extract process.

8. Target: Stop the Replicat process.
9. Source: Disable supplemental logging for the table to be modified by running DELETE TRANDATA.
10. Source: Make table DDL changes to the source table.
11. Target: Make table DDL changes to the target table.
12. Source: Re-enable supplemental logging by running ADD TRANDATA to the table(s) after the modifications have been performed.
13. Source: Start the Extract.
14. Target: Start the Replicat.
15. Source: Resume application data to the table or tables that were modified.

System Schemas for SQL Server

The following system schemas are not supported for Replication:

- sys
- cdc
- INFORMATION_SCHEMA

The guest schema is not supported using wildcard mappings, but is supported when explicitly listed, such as TABLE guest.employees.

Sybase

With Oracle GoldenGate for Sybase database, you can replicate data to and from supported Sybase versions or between a Sybase database and a database of another type. Oracle GoldenGate for Sybase supports data filtering, mapping, and transformation.

Prepare Database Users and Privileges for Sybase

Oracle GoldenGate requires a database user account. Create this account and assign privileges according to the following guidelines.

- 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, or operate as, the Oracle GoldenGate database user.
- Create a database user that is dedicated to Oracle GoldenGate. It can be the same user for all of the Oracle GoldenGate processes that must connect to a database:
 - Extract (source database)
 - Replicat (target database)
 - DEFGEN utility (source or target database)
- The Extract process requires permission to access the source database. Do one of the following:
 - Grant System Administrator privileges.
 - Assign a user name with replication_role . The command to grant replication role is either:

`sp_role 'grant', replication_role, Extract_user`

Or

```
use dbname grant role replication_role to Extract_user
```

Note

Specific DDL or DML operations may require the use of both `sa_role` and `replication_role`.

- The Replicat process requires connect and DML privileges on the target database.

Prepare Database Connection

Learn about configuring database connection, system, and parameter settings for Oracle GoldenGate for Sybase.

Oracle GoldenGate for Sybase connects to Sybase ASE databases using the client libraries available in the DB-Library package of the Sybase SDK FOR SAP ASE 16.0.3. Connections can be made either by using a listed connection and supplying the database server, port, database name and user information, or connections can be configured to use DSQUERY and provide the server name, database, and user information.

Both types of connections can be added from the **Administration Service** under the **DB Connections** menu in the Web UI, but using DSQUERY requires extra setup on the Oracle GoldenGate server.

Configure a DSQUERY Connection in Linux

Using DSQUERY connections requires connection details to be listed in an `interfaces` file which needs to be created and saved in the SDK FOR SAP ASE 16.0.3 installation folder, such as `/opt/sap`.

1. Connect as the user that installed the Sybase SDK, navigate to the Sybase SDK installation directory and create a new file, called `interfaces`, as shown. No file extension is required.

```
cd /opt/sap
vi interfaces
```

2. Add the following information to the `interfaces` file, editing the fields as per your environment.

```
servername
master tcp ether Host_IP_Address Database_Port_Number

query tcp ether Host_IP_Address Database_Port_Number

#Connections using SSL

#master tcp ether Host_IP_Address Database_Port_Number ssl="CN=host name"

#query tcp ether Host_IP_Address ssl="CN=<host name"
```


Note

It is important to follow the required layout of the connection information in the interfaces file.

For more information about the interfaces file, review the Sybase documentation available at:

<https://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.dc35823.1572/doc/html/san1334282782407.html>

The following rules are applicable to the interfaces file:

- Each SAP ASE has only one entry, although there may be multiple lines in the entry.
- Each line that follows the servername line must begin with a space or a tab
- Each element on the line must be separated by a single space.
- Each entry is separated by a blank line.

Example:

```
sybase_source
master tcp ether 10.0.0.100 5000

query tcp ether 10.0.0.100 5000
```

3. Save and close the file.
4. Once the entry has been created in the interfaces file, you can create a database connection from the Administration Service's web interface and select the **Server Name** option under **Connection Type**, listing the *servername* value of the interfaces file, and completing the remaining connection details.

Note

If, during deployment creation you did not set the LD_LIBRARY_PATH and SYBASE variables in the Environment Variables section of Step 3 of the Oracle GoldenGate Configuration Assistant, then you will need to manually add any missing variables, using the values for your Sybase SDK installation, else the database connections may fail.

Environment Variables can be added from the Service Manager's **Deployments** tab by expanding the deployment name and selecting **Configuration**. Any changes to the deployment's configuration require that the Administration Service be restarted for the changes to take effect.

Database Configuration

The Extract process makes calls directly to the Sybase Replication API on a source Sybase server. The source database on this server must be configured as follows to support data capture by Oracle GoldenGate.

- Because Extract uses the Sybase LTM to read the Sybase transaction log, it cannot run against a database configured with Sybase Replication Server. Only one process at a time can reserve a context that allows it to read the transaction log on the same database.
- Sybase Multisite availability leverages the Sybase Replication Server to replicate transactions to a Warm Standby and a target subscription database. Oracle GoldenGate for Sybase cannot capture from the primary Warm Standby but can capture from the Multisite availability target subscription database because SAP Sybase Rep Server is not in control of the Transaction Log for that database.
- The DSQUERY variable is used by Oracle GoldenGate to connect to the database using the server name. Set the DSQUERY variable to the server name that contains the database that Oracle GoldenGate will be using. In addition, set the SYBASE environment variable accordingly.
- The Extract process must be permitted to manage the secondary log truncation point. For more information, see [Initializing the Secondary Truncation Point](#).
- Configure the database page size to 4k, 8k, 16k, 32k, or larger. To use the `UPGRADECHECKPOINT table_name` command to update the checkpoint the target Sybase database, it must be configured to have a row size of more than 2K pages to be programmatically created; it will fail to upgrade the checkpoint if the target database is configured to be a 2K page size. This is valid only for Replicat.

Preparing Tables for Processing

The following table attributes must be addressed in an Oracle GoldenGate environment.

Disabling Triggers and Cascade Constraints

Disable triggers, cascade delete constraints, and cascade update constraints on target Sybase tables, or alter them to ignore changes made by the Oracle GoldenGate database user. Oracle GoldenGate replicates DML that results from a trigger or cascade constraint. If the same trigger or constraint gets activated on the target table, it becomes redundant because of the replicated version, and the database returns an error. Consider the following example, where the source tables are `emp_src` and `salary_src` and the target tables are `emp_targ` and `salary_targ`.

1. A delete is issued for `emp_src`.
2. It cascades a delete to `salary_src`.
3. Oracle GoldenGate sends both deletes to the target.
4. The parent delete arrives first and is applied to `emp_targ`.
5. The parent delete cascades a delete to `salary_targ`.
6. The cascaded delete from `salary_src` is applied to `salary_targ`.
7. The row cannot be located because it was already deleted in step 5.

To configure Replicat to disable target triggers at the start of its database session, take the following steps:

1. Assign the Replicat user the replication role.
2. Add the following parameter statement to the root level of the Replicat parameter file.

```
SQLEXEC "set triggers off"
```

Assigning Row Identifiers

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

Limiting Row Changes in Tables that do not Have a Key

If a target table has no primary key or unique key, duplicate rows can exist. It is possible for Oracle GoldenGate to update or delete too many rows in the target table, causing the source and target data to go out of synchronization without error messages to alert you. To limit the number of rows that are updated, use the `DBOPTIONS` parameter with the `LIMITROWS` option in the Replicat parameter file. `LIMITROWS` can increase the performance of Oracle GoldenGate on the target system because only one row is processed.

Replicating Encrypted Data

Oracle GoldenGate supports columns that are encrypted with a system-encrypted password, but not columns that are encrypted with a user-defined password. Check the tables from which you want to capture data against the following Oracle GoldenGate limitations:

- The table that contains the encrypted columns must have a primary or unique key.
- Columns that use encryption cannot be part of the primary key.

Encrypted columns are encrypted in the data files and in the log, so Extract must be configured to fetch the clear-text values from the database. To trigger this fetch, use the `FETCHCOLS` and `FETCHMODCOLS[EXCEPT]` options of the Extract `TABLE` parameter. `FETCHCOLS` forces a fetch of values that are not in the log, and `FETCHMODCOLS` or `FETCHMODCOLS[EXCEPT]` forces a fetch of values that are in the logs. Used together, these parameters ensure that the encrypted columns are always fetched from the database.

The following is an example of how to configure Extract to support the encryption. In this example, the encrypted column is `cardnum`.

```
TABLE ab.payments, FETCHCOLS (cardnum), FETCHMODCOLS (cardnum);
```

Preparing the Transaction Logs

To capture DML operations, Oracle GoldenGate reads the online logs. To ensure the continuity and integrity of Oracle GoldenGate processing, the logs must be configured as directed in the following sections:

Initializing the Secondary Truncation Point

Establish a secondary log truncation point on the source system prior to running the Oracle GoldenGate Extract process. Extract uses the secondary truncation point to identify data that remains to be processed.

To initialize the secondary truncation point, log on to the database as a user with `sa_role` privileges and then issue the following command:

```
dbcc settrunc( 'ltm', valid )
```

By default, Extract will manage the secondary truncation point once it is established. Do not permit Extract to be stopped any longer than necessary; otherwise the log could eventually fill up and the database will halt. The only way to resolve this problem is to disable the secondary truncation point and manage it outside of Oracle GoldenGate, and then purge the transaction

log. Data not yet processed by Extract will be lost, and you will have to resynchronize the source and target data.

To control how the secondary truncation point is managed, use the `TRANLOGOPTIONS` parameter. For more information, see *Reference for Oracle GoldenGate for Windows and UNIX*.

Sizing and Retaining the Logs

Retain enough log data on the source system so that Extract can start again from its checkpoints after you stop it or there is an unplanned outage. Extract must have access to the log that contains the start of the oldest uncommitted unit of work, and all logs thereafter. To determine where the Extract checkpoints are, use the `INFO EXTRACT` command. For more information about `INFO EXTRACT`, see *Reference for Oracle GoldenGate for Windows and UNIX*.

If data that Extract needs during processing is not retained, either in online or backup logs, one of the following corrective actions might be required:

- You might need to alter the Extract to capture from a later point in time for which log data is available (and accept possible data loss on the target). This can be achieved using the `ALTER EXTRACT` command.
- You might need to resynchronize the source and target tables, and then start the Oracle GoldenGate environment over again.

Make certain not to use backup or archive options that cause old archive files to be overwritten by new backups on the source system. New backups should be separate files with different names from older ones. This ensures that if Extract looks for a particular log, it will still exist, and it also ensures that the data is available in case it is needed for a support case.

Enabling Transaction Logging

Use the `ADD TRANDATA` command to mark each source table for replication. This command uses the Sybase `sp_setreptable` and `sp_setrepcol` system procedures. `ADD TRANDATA` is the recommended way to mark the tables, instead of using those procedures through the database interface, but the owner or the system administrator can use them if needed. For more information, see the Sybase documentation.

To mark tables for replication with `ADD TRANDATA`:

Run the `ADD TRANDATA` command for each table to be marked. The syntax of the `ADD TRANDATA` command is:

```
ADD TRANDATA SCHEMA.TABLE LOBSNEVER | LOBSALWAYS | LOBSALWAYSNOINDEX |  
LOBSIFCHANGED
```

`LOBSNEVER | LOBSALWAYS | LOBSALWAYSNOINDEX | LOBSIFCHANGED` control whether LOB data is never propagated, only propagated if changed (the default), or always propagated. The `LOBSIFCHANGED` is the default value if no LOB option is specified. The `ADD TRANDATA` command will overwrite the LOB replication setting that is currently set for the table.

Note

Some `ADD TRANDATA` options enable the `ALWAYS_REPLICATE` option of `sp_setrepcol`. If a LOB column contains a NULL value, and then another column in the table gets updated (but not the LOB), that LOB will not be captured even though `ALWAYS_REPLICATE` is enabled.

Sybase: Supported Data Types, Objects, and Operations

This section contains information on supported data types, objects, and operations for Oracle GoldenGate for Sybase.

Supported Sybase Data Types

Integers

- BIGINT
- BIT
- DECIMAL
- INT (signed and unsigned)
- TINYINT (signed and unsigned)
- NUMERIC
- SMALLINT (signed and unsigned)

Limitations of Support

- NUMERIC and DECIMAL (fixed-point) are supported with no integrity loss when moving data to a target column of the same data type without involving calculations or transformation. When calculations or transformation must be performed, Oracle GoldenGate supports a maximum value of a signed long integer (32-bits).
- BIT is supported for automatic mapping between Sybase databases. To move BIT data between Sybase and another database type, Oracle GoldenGate treats BIT data as binary. In this case, the following are required:
 - The BIT column must be mapped to the corresponding source or target column with a COLMAP clause in a TABLE or MAP statement.
- For the Sybase 15.7 and above releases, these data types cannot be replicated:
 - BIGINT (as a key column)
 - BIGDATETIME
 - BIGTIME
- When replicating TINYINT and Extract is not in the same version of Replicat, you will need to create a sourcedef and/or targetdef file even if you are replicating between identical Sybase versions.

See also [Non-Supported Sybase Data Types](#).

Floating-Point Numbers

- DOUBLE
- FLOAT
- REAL

Limitations of Support

The support of 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.

Character Data

- CHAR
- NCHAR
- NVARCHAR
- VARCHAR
- UNICHAR
- UNIVARCHAR

Limitations of Support

- These data types are supported to the maximum length supported by the database, this being the maximum page size.
- Fetching NVARCHAR replication results using the Sybase char_length or data_length functions when a Sybase database is the target and the source is a heterogeneous database and you replicate from the source to the target may result in a data integrity issue. This occurs when you use a Sybase release earlier than Adaptive Server Enterprise 15.5 for Windows x64 platform EBF 21262: 15.5 ESD #5.3.

Dates and Timestamps

- BIGDATETIME
- BIGTIME
- DATE
- DATETIME
- SMALLDATETIME
- TIME

Limitations of Support

- 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 time zone, conversion may add or subtract hours, which can cause the timestamp to exceed the lower or upper supported limit.
- Oracle GoldenGate does not support negative dates.

Large Objects

- BINARY
- IMAGE
- TEXT
- UNITEXT
- VARBINARY

Limitations of Support

- TEXT, UNITEXT and IMAGE are supported up to 2 GB in length.

- Large objects that are replicated from other databases (such as Oracle BLOB and CLOB) can be mapped to Sybase CHAR, VARCHAR, BINARY, and VARBINARY columns. To prevent Replicat from abending if the replicated large object is bigger than the size of the target column, use the DBOPTIONS parameter with the ALLOWLOBDATATRUNCATE option in the Replicat parameter file. For more information, see Reference for Oracle GoldenGate for Windows and UNIX.
- To move data to a Sybase target from a source database that permits empty LOB columns, use the DBOPTIONS parameter with the EMPTYLOBSTRING option in the Replicat parameter file. This parameter accepts a string value and prevents Replicat from setting the target column to NULL, which is not permitted by Sybase. For more information, see Reference for Oracle GoldenGate for Windows and UNIX.
- When a source table contains multiple identical rows, it can cause LOB inconsistencies in the target table. This occurs when the source table lacks a primary key or other unique row identifier. The rows are inserted by Replicat on the target, but if the LOB data is updated in a subsequent source operation, it will only be replicated to the first row that was inserted on the target.
- Do not use NOT NULL constraints on the in-row LOB column. If you want to use NOT NULL constraints, use them on the off-row LOB column.
- If you need to fetch the in-row LOB data directly from the table you must use FETCHCOLS/FETCHMODCOLS.
- Oracle GoldenGate for Sybase 15.7 and above does not support the in-row LOB column replication (however, it can still push the data into the in-row LOB column at in the Replicat database). This means tables included in the replication cannot have any in-row LOB columns. Oracle GoldenGate will abend if any replication table includes an in-row LOB column. If you need in-row LOB support, contact Oracle Support for further information.

Money Types

- MONEY
- SMALLMONEY

Limitations of Support

Money data types are supported with no integrity loss when moving data to a target column of the same data type without involving calculations or transformation. When calculations or transformation must be performed, Oracle GoldenGate supports a maximum value of a signed long integer (32-bits).

IDENTITY Type

The IDENTITY data type is supported for replication in one direction only, but not for a bi-directional configuration.

text, image, and unitext Data Types

With the Sybase 15.7 version, the LOB text, image, and unitext data types are now supported in BATCHSQL mode. The data length of the LOB is confined to 4K. If the records that contain LOB columns and the size exceeds more than 4K, then those records are excluded from the batches and are executed one at a time. The LOB columns in are now bound, while in previous Sybase version (15.5 or 15.0) the LOBs were not bound. You can use the older behavior by using the DBOPTIONS LEGACYLOBREPLICATION parameter. This support is only applicable to Replicat running on Sybase version 15.7 and later.

User-Defined Data Types

User-defined data types are fully supported.

Non-Supported Sybase Data Types

This section lists the Sybase data types that Oracle GoldenGate does not support.

- The `TIMESTAMP` data is not supported. Timestamp columns data is captured though the data cannot be applied to the Sybase timestamp column due to a database limitation. The database populates this column automatically once that corresponding row is inserted or updated. To exclude timestamp columns from being captured by Oracle GoldenGate, use the `COLSEXCEPT` option of the `TABLE` parameter. Because the system generates the timestamps, the source and target values will be different.
- The Java rowobject data type is not supported.

Supported Operations and Objects for Sybase

This section lists the data operations and database objects that Oracle GoldenGate supports.

- The extraction and replication of insert, update, and delete operations on Sybase tables that contain rows of up to 512 KB in length.
- The maximum number of columns and the maximum column size per table that is supported by the database.
- Deferred inserts, deferred indirect inserts, deferred updates, and deferred deletes. It is possible that the use of deferred updates could cause primary key constraint violations for the affected SQL on the target. If these errors occur, use the Replicat parameter `HANDLECOLLISIONS`.
- `TRUNCATE TABLE` if the names of the affected tables are unique across all schemas. If the table names are not unique across all schemas, use the `IGNORETRUNCATES` parameter for those tables to prevent Replicat from abending.
- `GETTRUNCATES` and `IGNORETRUNCATES` by Extract and Replicat.
- Data that is encrypted with a system-encrypted password.
- Array fetching during initial loads, as controlled by the `FETCHBATCHSIZE` parameter.
- The `BATCHSQL` Replicat feature on ASE 15.7 SP110 and later on the following platforms:
 - AIX
 - Linux x64
 - Sun Solaris SPARC
 - Sun Solaris x64
 - Windows x64

In certain scenarios, the `CS_NUMERIC` and `CS_DECIMAL` data types are not supported by BatchSQL because of a bug in the Sybase specific CT Library. LOB replication is supported in BatchSql mode for Sybase database version 157 SP110 onward. This will improve the LOB replication performance. It is restricted to 16384 bytes of LOB data that means if LOB data is more than 16384 bytes, the data would not be processed through `BATCHSQL` mode instead the mode switched to Normal.

- Limitations on Computed Columns support are as follows:
 - Fully supports persisted computed columns. The change values are present in the transaction log and can be captured to the trail.
 - You cannot use `NOT NULL` constraints on in-row LOB columns. If you need to use `NOT NULL` constraints, do so only with off-row LOB columns.

- Tables with non-persisted computed columns, but does not capture change data for these columns because the database does not write it to the transaction log. To replicate data for non-persisted computed columns, use the `FETCHCOLS` or `FETCHMODCOLS` option of the `TABLE` parameter to fetch the column data from the table. Keep in mind that there can be discrepancies caused by differences in data values between when the column was changed in the database and when Extract fetches the data for the transaction record that is being processed.
- Replicat does not apply DML to any computed column, even if the data for that column is in the trail, because the database does not permit DML on that type of column. Data from a source persisted computed column, or from a fetched non-persisted column, can be applied to a target column that is not a computed column.
- In an initial load, all of the data is selected directly from the source tables, not the transaction log. Therefore, in an initial load, data values for all columns, including non-persisted computed columns, gets written to the trail or sent to the target, depending on the method that is being used. As when applying change data, however, Replicat does not apply initial load data to computed columns, because the database does not permit DML on that type of column.
- Persisted computed column that is defined as a key column, an index column, or that is part of a `KEYCOLS` clause in a `TABLE` or `MAP` statement are not used. If a unique key or index includes a computed column and Oracle GoldenGate must use that key, the computed column will be ignored. Additionally, if a unique key or index contains a computed column and is the only unique identifier on the table, all of the columns are used except the computed column as an identifier to find the target row. Thus, the presence of a computed column in a key or index affects data integrity if the remaining columns do not enforce uniqueness. Sybase does not support non-persisted computed columns as part of a key so neither does Oracle GoldenGate.
- To support `TRUNCATE TABLE`, all table names should be unique across all schemas within a database. This rule applies to Extract and Replicat.
- Limitations on Automatic Heartbeat Table support are as follows:
 - Heartbeat frequency is accepted in minutes and should be an integer between 1 and 133 (both inclusive).
 - Data truncation occurs with a Replicat abend when it exceeds more than 1500 characters for the `incoming_routing_path` and `outgoing_routing_path` of the `GG_HEARTBEAT_SEED`, `GG_HEARTBEAT`, and `GG_HEARTBEAT_HISTORY` tables. The `incoming_routing_path` and `outgoing_routing_path` size of these table is set to 1500 characters in ASCII and is a 500 max bytes in multibyte characters. Ensure that the incoming and outgoing routing path strings are within the specified limit.
 - Sybase job scheduler must be configured on the ASE server prior to running Oracle GoldenGate heartbeat functionality.
 - For heartbeat table functionality to operate correctly, the login user must have the `replication_role`, `js_admin_role`, `js_user_role` roles.

Non-Supported Operations and Objects for Sybase

This section lists the data operations and database objects that Oracle GoldenGate does not support.

- Replication of materialized views or precomputed result sets is not supported. Therefore, you cannot add trandata for materialized views. See the [Sybase ASE documentation](#) for details.
- Data that is encrypted with a user-defined password.

- Extraction or replication of DDL (data definition language) operations.
- Multi-Extract configuration. Only one Extract can reserve a context to read the Sybase transaction logs.
- Because SHOWSYNTAX is supported in the DYNSQL mode, NODYNSQL is deprecated.
- Table names that contain data with an underscore followed by some characters then a space (for example, 'zzz_j ') is not supported. Oracle GoldenGate cannot process records containing this type of character string with GGSCI, DEFGEN, EXTRACT, or REPLICAT. Additionally, this type of data cannot be used with Oracle GoldenGate wildcard (*). If you do have this type of data in your table name, you must drop this kind of table name from your database, and then they restart the application to process and respect Oracle GoldenGate wildcard.

Teradata

With Oracle GoldenGate for Teradata, you can deliver initial load and transactional data from other supported Oracle GoldenGate sources, such as an Oracle database.

Oracle GoldenGate for Teradata supports data filtering, mapping, and transformations unless noted otherwise in this documentation.

Prepare Database Users and Privileges for Teradata

Requirements for the database user for Oracle GoldenGate processes are as follows:

- Create a database user that is dedicated to Oracle GoldenGate. It can be the same user for all of the Oracle GoldenGate processes that must connect to a database.
- Keep a record of the database users credentials to set up database connection from Oracle GoldenGate. These credentials are required when setting up the USERIDALIAS for database login (DBLOGIN) from Oracle GoldenGate.
- 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, the Oracle GoldenGate database user.
- For Oracle GoldenGate to replicate to a target Teradata database, grant SELECT, INSERT, UPDATE, and DELETE on all of the target tables to the Replicat database user.

Prepare Database Connection

Learn about configuring database connections for Oracle GoldenGate for Teradata.

Oracle GoldenGate for Teradata connects to Teradata databases using an ODBC Driver for Teradata and connection information provided in a Data Source Name (DSN). DSN connectivity requires connection details be listed in an `odbc.ini` file and then referencing the DSN value when creating database connections within the Oracle GoldenGate WebUI or Admin Client.

On the Oracle GoldenGate server, install and configure a supported ODBC Driver for Teradata and create a DSN entry to be used for Oracle GoldenGate connectivity to a target Teradata system.

Review the Oracle GoldenGate [certification matrix](#) for this release of Oracle GoldenGate to determine which version of the Teradata TTU is supported. Download and install that version of the ODBC Driver for Teradata available at <https://downloads.teradata.com/>.

Once the DSN entries have been created through the steps mentioned in the [Configure a DSN Connection in Linux](#) topic, proceed to the [Add Database Connections](#) topic to know about how to create database connections.

Configure a DSN Connection in Linux

To create a DSN entry for Oracle GoldenGate processes, perform the following steps to first, add a new environment variable for the Oracle GoldenGate for Teradata deployment, update an existing environment variable, and then create an `odbc.ini` file to store the connection attributes:

1. Log in to the Service Manager web interface.
2. From the left navigation pane, click **Deployments** and then select the Oracle GoldenGate Teradata deployment. This displays the available settings for the deployment.
3. Click **Configuration** and then click the plus sign (+) displayed next to **Environment Variables**.
4. Provide the following information in the two available fields:

Environment Variable Name = `ODBCINI`

Environment Variable Value = `Teradata deployment path/etc/conf/ogg/odbc.ini`

Note

You can choose to create the `odbc.ini` file in any location that you prefer, but it must be referenced correctly in the `ODBCINI` environment variable for the deployment.

Figure 7-2 Add Environment Variable

Add Environment Variable

×

The Deployment contains the following Environment Variable Settings. Environment Variables can be added, modified, or deleted.

Environment Variable Name	Environment Variable Value
ODBCINI	/home/oracle/ogg/deployments/teradata/etc/conf/ogg/odbc.ini

Required

5. Click **Submit** to create the new variable.
6. Next, edit the existing `LD_LIBRARY_PATH` environment variable by clicking the pencil icon under **Actions** for the `LD_LIBRARY_PATH` entry.
7. Add the `lib` path of the ODBC Driver for Teradata to the existing `LD_LIBRARY_PATH` value, such as: `/opt/teradata/client/17.20/odbc_64/lib`
8. Click **Submit** to update the variable and then restart the deployment from the Service Manager **Deployments** pane, for the changes to take effect.
9. In the directory that you set for the `ODBCINI` environment variable, create an `odbc.ini` file and add Teradata database connections to this file, using the following minimum information and example. For more details about available connection options, review the [ODBC Driver for Teradata User Guide](#).

Example of `odbc.ini` file

```
[ODBC]
InstallDir=/opt/teradata/client/17.20/odbc_64
[ODBC Data Sources]
DSN Name=Teradata Database ODBC Driver 17.20
[DSN Name]
Driver=/opt/teradata/client/17.20/odbc_64/lib/tdataodbc_sb64.so
```

10. Save and close the `odbc.ini` file.

Preventing Multiple Connections

By default, the Replicat processes create a new connection for catalog queries. You can prevent this extra connection by using the `DBOPTIONS` parameter with the `NOCATALOGCONNECT` option.

Preparing Tables for Processing

The following table attributes must be addressed in an Oracle GoldenGate environment for Teradata.

Disabling Triggers and Cascade Constraints

Disable triggers, cascade delete constraints, and cascade update constraints on target Teradata tables if Oracle GoldenGate replicates DML from a source that occurred from a trigger or cascade constraint. If the same trigger or constraint gets activated on the target table, it becomes redundant due to the replicated source data, and the database will return an error. Consider the following example, where the source tables are `emp_src` and `salary_src` and the target tables are `emp_targ` and `salary_targ`.

1. A delete is issued for `emp_src`.
2. It cascades a delete to `salary_src`.
3. Oracle GoldenGate sends both deletes to the target.
4. The parent delete arrives first and is applied to `emp_targ`.
5. The parent delete cascades a delete to `salary_targ`.
6. The cascaded delete from `salary_src` is applied to `salary_targ`.
7. The row cannot be located because it was already deleted in step 5.

Ensuring Row Uniqueness for Tables

Oracle GoldenGate requires some form of 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:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.

3. 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 those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration.

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.

4. If a table does not have an appropriate key, or if you prefer that the existing key(s) are not 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. See *TABLE | MAP in Parameters and Functions Reference for Oracle GoldenGate*.

Using KEYCOLS to Specify a Custom Key

If a table does not have an applicable row identifier, or if you prefer that identifiers are not used, you can define a substitute key, providing that 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 overrides any existing primary or unique key that Oracle GoldenGate finds.

Handling Massive Update and Delete Operations

Operations that update or delete a large number of rows generate discrete updates and deletes for each row on the subscriber database. This could cause a lock manager overflow on the Teradata subscriber system, and thus terminate the Replicat process.

To avoid these errors, temporarily suspend replication for these operations and then perform them manually on the source and target systems. To suspend replication, use the following command, which suspends replication for that session only. The operations of other sessions on that table are replicated normally.

```
set session override replication on;  
commit;
```

Teradata: Supported Data Types, Objects, and Operations

This section contains information on supported data types, objects, and operations for Oracle GoldenGate on Teradata.

Supported Teradata Data Types

The following are the Teradata data types that Oracle GoldenGate supports. Any limitations or conditions that apply, are mentioned after the list of supported datatypes.

- BLOB

- BYTE
- VARBYTE
- BIGINT
- BYTEINT
- DATE
- DECIMAL
- DOUBLE PRECISION
- FLOAT
- INTEGER
- NUMBER
- NUMERIC
- REAL
- SMALLINT
- TIME
- TIMESTAMP
- INTERVAL
- INTERVAL DAY
- INTERVAL DAY TO HOUR
- INTERVAL DAY TO MINUTE
- INTERVAL DAY TO SECOND
- INTERVAL HOUR
- INTERVAL HOUR TO MINUTE
- INTERVAL HOUR TO SECOND
- INTERVAL MINUTE
- INTERVAL MINUTE TO SECOND
- INTERVAL MONTH
- INTERVAL SECOND
- INTERVAL YEAR
- INTERVAL YEAR TO MONTH
- CHAR
- CLOB
- CHAR VARYING
- LONG VARCHAR
- VARCHAR
- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC

The REAL and DOUBLE data types support the NaN and Infinity/-Infinity special numeric values.

For more information on data types by databases that support special values, refer to [Special Values Supported by Database Data Types](#).

Limitations of Support for Data Types

Here are the details of conditions and limitations of support for data types.

Date Data Types

The following are the limitations of support for date data types:

- TIME with TIMESZONE, TIMESTAMP with TIMEZONE, and INTERVAL are not supported from a different type of source database to Teradata.
- 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 time zone, conversion may add or subtract hours, which can cause the timestamp to exceed the lower or upper supported limit.

IDENTITY Columns

IDENTITY must be configured as GENERATED BY DEFAULT AS IDENTITY on the target to enable the correct value to be inserted by Replicat.

Large Object Data Types

The following is the limitation of support for large object data types:

- Enable the UseNativeLOBSupport flag in the ODBC configuration file. See the Teradata ODBC documentation.

Numeric Data Types

When replicating the numeric data types to Teradata, truncation can occur if the source database supports a higher precision than Teradata does.

The support of 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.

Multi-byte Character Data

For multi-byte character data, the following conditions and limitations are applicable:

- Filtering, mapping, and transformation for multi-byte data types is not supported.
- Set the ODBC driver to the UTF-16 character set in the initialization file.
- When creating a Replicat, use the NODBCCKPOINT option with the ADD REPLICAT command. The Replicat database check-pointing feature does not support an ODBC driver that is set to the UTF-16 character set. Checkpoints will be maintained in the checkpoint file on disk.

Supported Objects and Operations for Teradata

This section lists the data operations and database objects that Oracle GoldenGate supports.

- Oracle GoldenGate for Teradata supports delivery (Replicat) of DML transactions to Teradata.
- Oracle GoldenGate supports the maximum number of columns per table that is supported by the database.
- Truncating operations are supported with the use of the GETTRUNCATES parameter.
- Limitations on Automatic Heartbeat Table support are as follows:
 - The ALTER HEARTBEATTABLE command is not supported and if used is ignored.
 - The ADD HEARTBEATTABLE command with the FREQUENCY, PURGE_FREQUENCY, or RETENTION_TIME option is not supported. When any of these options are specified with the ADD HEARTBEATTABLE command, a warning is displayed that the option is ignored.
 - Since Teradata does not have any internal event/job schedulers, automatic purging of heartbeat history data does not occur. You need to explicitly delete or truncate records periodically from the heartbeat history table.

Non-Supported Operations for Teradata

This section lists the operations that Oracle GoldenGate for Teradata does not support.

- Extract (capture) is not supported for Teradata
- DDL replication

Yugabyte

With Oracle GoldenGate for Yugabyte, you can perform capture and delivery from PostgreSQL source to Yugabyte target database on a Linux platform.

Prepare Database Connection

Oracle GoldenGate for Yugabyte (PostgreSQL variant) connects to Yugabyte databases using a pre-packaged ODBC driver. Connections can be established using a Data Source Name (DSN) or using a direct connection and supplying the database server host, port, database, and other information.

Using DSN connections requires connection details to be listed in an `odbc.ini` file, while using direct entries are entered manually when adding a database connection from the web interface or from the Admin Client.

Note

PgBouncer is not supported for Oracle GoldenGate connections.

Note

Oracle GoldenGate does not support connections to Yugabyte that use `Pgpool`.

After performing the steps given below to create the DSN entries or you plan to use direct connections, proceed to the [Add Database Connections](#) topic to know how to create database connections.

Set up a DSN-based Connection

To create a DSN connection for Oracle GoldenGate processes, you will first need to add an environment variable for the Oracle GoldenGate for Yugabyte deployment and then create an `odbc.ini` file to store the connection attributes.

1. Log in to the Service Manager web interface.
2. From the left navigation pane, click **Deployments** and then select the deployment name for Yugabyte. This expands the various settings for the deployment.
3. Click **Configuration** and then click the plus sign (+) next to **Environment Variables**.
4. Provide the following information in the two available fields. For the **Environment Variable Value** field, ensure it is the absolute path of the `odbc.ini` file:

Environment Variable Name = `ODBCINI`
Environment Variable Value = `/ogg/deployment/yugabyte/odbc.ini`
5. Click **Submit** to create the new variable and then restart the deployment from the **Deployments** pane, for the changes to take effect.
6. In the Oracle GoldenGate installation's deployment folder, manually create an `odbc.ini` file and add data sources in this file.

Use the following minimum settings when creating the DSN file:

- **Data Source Name** – A user defined name of a source or target database connection that will be referenced by Oracle GoldenGate processes, such as Extract or Replicat. DSN names are allowed up to 32 alpha-numeric characters in length, and can include only underscore (`_`) and dash (`-`) from special characters.
- **IANAAppCodePage=4** – Is the default setting but can be modified according to the guidance specified on the https://docs.progress.com/bundle/datadirect-connect-odbc-71/page/IANAAppCodePage_9.html#IANAAppCodePage_9 page when the database character set is not Unicode.
- **InstallDir** – Is the name of the Oracle GoldenGate PostgreSQL wire protocol driver path, and can use a relative path, like: `InstallDir = ./datadirect`
- **Driver**: Is the name of the Oracle GoldenGate PostgreSQL Wire Protocol driver file, and can use a relative path, like: `Driver= ./datadirect/lib/ggsql25.so`.
- **Database**: This is the name of the source or target database.
- **Hostname**: This is the database host IP address or host name.
- **PortNumber**: This is the listening port of the database.
- Oracle GoldenGate for Yugabyte support multiple hosts and ports for Yugabyte connections. This feature is useful in the Oracle GoldenGate High Availability (HA) setup. In a Yugabyte HA setup there is one primary and multiple standby servers. You can set up the connection string format to specify multiple hosts:

Connection String	Description
<i>db-user@host1,host2,host3/db-name</i>	The connections would be attempted using default Yugabyte database port 5433 for all the hosts (host1, host2, host3).
<i>db-user@host1:1234,host2:2345,host3:3456/db-name</i>	The connections would be attempted using the ports associated with each host specified in the connection string.
<i>db-user@host1,host2,host3:1234/db-name</i>	All hosts using the same port number (1234) on all the hosts (host1, host2, host3). The connections would be attempted using the port 1234 for all the hosts (host1, host2, host3).
<i>db-user@host1,host2,host3:1234/db-name</i>	All hosts using the same port number (1234) on all the hosts (host1, host2, host3). The connections would be attempted using the port 1234 for all the hosts.
<i>db-user@host1:3456,host2,host3:1234/db-name</i>	If the connection string is specified in other formats where the port number is specified for some hosts while the other hosts are specified without any ports, then the last or right-most port specification will be used as the port number for all the hosts which do not have a port number specified in the connection string. As shown in the connection string: <i>db-user@host1:3456,host2,host3:1234/db-name</i> The port number for <i>host2</i> would be 1234, which is the right-most port number specified in the connection string.

- You can also provide a LogonID and Password for the Extract or Replicat user, but these will be stored in clear text. It is recommended to leave these fields out of the DSN and instead store them in the Oracle GoldenGate wallet as a credential alias, and reference them with the USERIDALIAS parameter in Extract and Replicat.

The following is an example `odbc.ini` file with two DSN entries. The Data Source names used in the example below are `PG_src` and `PG_tgt`.

```
#[ODBC Data Sources]

[ODBC]
IANAAppCodePage=4
InstallDir=/datadirect

[PG_src]
Driver= ./datadirect/lib/ggpsql25.so
```

```
Description=Oracle GoldenGate PostgreSQL Wire Protocol
Database=sourcedb
HostName=remotehost
PortNumber=5432
```

```
[PG_tgt]
Driver=./datadirect/lib/ggpgsql25.so
Description=Oracle GoldenGate PostgreSQL Wire Protocol
Database=targetdb
HostName=remotehost
PortNumber=5432
```

1. Save and close the `odbc.ini` file.
2. To set up the database connection from Oracle GoldenGate for a Yugabyte deployment, see [Add Database Connections](#).

Connecting to a FIPS-enabled Yugabyte System with Version 14 or Lower

When the Oracle GoldenGate Extract is run from a Federal Information Processing Standards (FIPS) enabled system installed with Yugabyte database lower than version 14, it generates the following error:

```
ERROR OGG-25359 Could not connect to server with database 'postgres', host
'localhost', port '5432' and user name 'postgres'. Error Message: connection to
server at "localhost" (:::1), port 5432 failed: could not encrypt password:
disabled for FIPSfe_sendauth: error sending password authentication.
```

The encryption algorithm `md5` is the default encryption algorithm on Yugabyte database version lower than 14 and causes the Extract to abend with an error.

To run Extract on a FIPS-enabled system running Yugabyte database version lower than 14, perform the following steps:

1. Modify the `postgresql.conf` file to set the `password_encryption` option to `scram-sha-256`.
2. Modify the `pg_hba.conf` file to set the `Method` option to `scram-sha-256`, as `md5` is not supported on a FIPS-enabled system. However, this is an optional step.

The password for the database user that is used by Oracle GoldenGate Extract, must be re-generated or modified if the database user has already been created, after the `password_encryption` option is set to `scram-sha-256`. You can use the same password to be regenerated.

For example, if the database user, named `admin` uses the password as `password123`, then the same password can be regenerated using the `scram-sha-256` encryption.

Configuring SSL Support for Yugabyte

SSL can be enabled by setting the configuration parameter `SSL` to `on` in the Yugabyte configuration file (`$PGDATA/postgresql.conf`). If SSL is enabled, the corresponding `hostssl` entry must be present or added in the `pg_hba.conf` file.

When SSL is enabled, Oracle GoldenGate uses the root certificate, root certification revocation list (CRL), server client certificate, and key from the default locations, as shown in the following snippet:

```
~/.yugabyte/root.crt  
~/.yugabyte/root.crl  
~/.yugabyte/yugabyte.crt  
~/.yugabyte/yugabyte.key
```

You need to create the desired entities from this list, and store them in appropriate locations.

If the SSL configuration is setup using non-default locations, then the following environment variables should be set up as per the environment.

```
PGSSLROOTCERT  
PGSSLCRL  
PGSSLCERT  
PGSSLKEY
```

Changes required in `odbc.ini` file

The SSL support can be enabled by setting the `EncryptionMethod` DSN attribute to 1 or 6 in the `$ODBCINI` file.

If set to 0 (No Encryption), data is not encrypted.

If set to 1 (SSL), data is encrypted using the SSL protocols specified in the `Crypto Protocol Version` connection option. If the specified encryption method is not supported by the database server, the connection fails and the driver returns an error.

If set to 6 (RequestSSL), the login request and data are encrypted using SSL if the server is configured for SSL. If the server is not configured for SSL, an unencrypted connection is established. The SSL protocol used is determined by the setting of the `Crypto Protocol Version` connection option.

If the database server/client certificates also need to be validated, then the corresponding `KeyStore` file needs to be created and the following ODBC DSN attributes should be setup accordingly in the `odbc.ini` file.

```
KeyStore=path to .p12 keystore file KeyStorePassword=keystore-passwd  
TrustStore=path to root certificate ValidateServerCertificate=1
```

However, the `KeyStore` file is not mandatory for validation of client certificates (by the server) when `SSLCert` and `SSLKey` are already provided.

Note

Azure Database for Yugabyte defaults to enforce SSL connections. To adhere to this requirement, perform the requirements listed here, or optionally, you can disable enforcing SSL connections from the Connection security settings of the database instance using the Microsoft Azure Portal.

Yugabyte: Supported Objects and Operations

The following list describes the supported objects and operations for Oracle GoldenGate for Yugabyte:

- Oracle GoldenGate Extract supports yboutput plugin.
- The Replica Identity CHANGE is the native setting for Yugabyte and is recommended when using the yboutput plugin, as it provides better performance compared to other Replica Identity types.
- Bidirectional replication is supported with yboutput plugin. When using the yboutput plugin for bidirectional replication, the Oracle GoldenGate checkpoint table must be created and included in the publication list before starting the CDC Extract process. Additionally, the replication slot should be created only after the checkpoint table has been created and added to the publication.
- Bidirectional replication is supported with test_decoding plugin. All the tables including the checkpoint tables must be created before registering any Extract when using test_decoding plugin. In case of PGOUTPUT plugin, the checkpoint table along with all other interested tables must be included in the publication before Extract registration.

Note

Do not drop any table including checkpoint tables.

- Multiple truncate operations are supported in pgoutput/yboutput plugin only.
- Register command enhanced with the pgplugintype option:
 - The pgplugintype option is available in both the web interface and Admin Client under the REGISTER EXTRACT command. This option allows you to choose either test_decoding or pgoutput as the logical decoding plugin.
 - When you specify pgplugintype pgoutput, then the underlying module automatically uses the yboutput plugin to register the Extract process.
 - If you don't specify any pgplugintype, then the CDC Extract registers using the test_decoding plugin, by default.
 - To explicitly register with the yboutput plugin in Yugabyte, you must specify the pgoutput plugin type.

Syntax:

```
REGISTER EXTRACT group_name PGPLUGINTYPE plugin_option;
```

Example:

```
REGISTER EXTRACT ext_yb PGPLUGINTYPE pgoutput;
```

- The STREAMINGOPTIONS parameter has been introduced under TRANLOGOPTIONS. This parameter allows you to specify the publication_name for the replication start command.

Example:

```
TRANLOGOPTIONS STREAMINGOPTIONS  
(publication_names="my*123,my_publication2,123 pub,my_publication");)
```

The following guidelines and restrictions must be considered when using this parameter:

- The publication(s) must be created before registering the Extract process.
Publication names must not contain special characters including Comma (,) Single quote (') Double quote (").
- Multiple publication names can be specified as a comma-separated list within publication_names. For example:

```
TRANLOGOPTIONS STREAMINGOPTIONS  
(publication_names="my*123,my_publication2,123 pub,my_publication");)
```

The final publication name in the list must end with a semicolon (;)

Yugabyte Limitations

Consider the following limitations for the Yugabyte database, while setting it up for capture and delivery:

- **Missing large LOB transactions:** When an LOB (Large Object) has a size between 200 MB and 400 MB, the entire transaction, whether an INSERT or UPDATE, may be missing from the CDC stream.
- **Data mismatch in downstream due to multiple updates in the same transaction:** When multiple updates occur within the same transaction, the CDCSDK module in Yugabyte may fail to send the correct AFTER image if the table's replica identity is set to FULL or DEFAULT. This can result in data discrepancies for the AFTER image in the downstream system.
- **Replication connection terminates after DDL on replicated tables (test_decoding Plugin):** When using the test_decoding plugin, all tables in the database are automatically included in the polling list, by default. If a DDL operation (such as a table drop) is performed on any table included in this list, the server terminates the replication connection. However, dropping tables that are part of the polling list is not supported with the test_decoding plugin, so, if you require this flexibility, you should switch to the yboutput plugin.

Note

Before dropping a table, ensure that it is not part of any publication to avoid replication errors or connection interruptions. Similarly, do not drop the Oracle GoldenGate checkpoint table as it will also result in the same error.

- **Replication slot expiration due to extended inactivity:** Replication slots retain certain resources to prevent garbage collection of WAL (Write-Ahead Log) records required for replication. If a slot remains inactive for an extended period, it is marked as expired, and the retained resources are released. After expiry, the replication slot becomes unusable and must be recreated.

Default retention window:

- 8 hours starting from version 2024.2.1
- 4 hours in versions prior to 2024.2.1

This is a known issue as mentioned on the Yugabyte site:

<https://github.com/yugabyte/yugabyte-db/issues/27766>

Recommended Actions to Prevent Expiration:

- Enable heartbeat functionality to ensure continuous WAL activity and prevent long idle periods.
- Configure **autostart** or **autorestart** options for the CDC process so that it automatically reconnects to the CDCSDK module, minimizing downtime and avoiding slot expiration.
- **Tables eligible for publication only:** Persistent base tables and partitioned tables can be included in a publication. The following object types cannot be part of a publication:
 - Temporary tables
 - Unlogged tables
 - Foreign tables
 - Materialized views
 - Regular views
- **Publication deletion impact on CDC Extract:** If a publication is deleted while the CDC Extract is still running, there is a risk of replication slot corruption, causing it to stop functioning. In such cases, the affected replication slot becomes unusable, and simply recreating the same publication will not resolve the issue. There is no solution to resume on the existing stream and only way is to create a new publication and associate it with a new replication slot.
- **Limitations with Timezone (TZ):** The WAL sender process always transmits committed timestamps in UTC. If the Oracle GoldenGate session timezone differs from UTC, then discrepancies may occur while positioning by timestamp in the CDC Extract process. To prevent this issue, you must set the environment variable TZ=UTC at the deployment layer. This limitation applies specifically to the yboutput plugin.
- pgoutput plugin is not supported for Yugabyte.
- Globalization is not supported with yboutput plugin
- **Primary key on table:** Only tables with a primary key are supported for CDC streaming in Yugabyte. The table that does not have the primary key, will not be captured by Oracle GoldenGate CDC Extract.
- **Primary Key updates in Yugabyte:** A PK UPDATE in Yugabyte will be logged as DELETE followed by INSERT. Oracle GoldenGate Extract will show the statistics for a PK UPDATE as DELETE and INSERT only. Although the statistics will not be shown as update operation in the Extract statistics, but there will not be any discrepancy in the data on source and target.

- **Replica identity behavior:** Replica identity of tables present at the time of slot creation cannot be changed after the slot is created. However, replica identity of all new tables created after slot creation can be set using `ysql_yb_default_replica_identity`, `cdcsdk_publication_list_refresh_interval_secs` and `cdsdk_enable_dynamic_table_support` tserver flag. These flags need to be set in the master as well as t-server before new tables are created.

By default, `cdcsdk_publication_list_refresh_interval_secs` is set to 900 sec, which is 15 minutes. You need to set this as per your requirement. Decreasing its value will reduce the newly created table to be refreshed in the CDCSDK module quickly, which will help Oracle GoldenGate capture any records for these new tables. The following example shows the flags that are set for dynamically created tables after the SLOT is created:

```
ysql_yb_default_replica_identity=FULL,cdcsdk_publication_list_refresh_interval_secs=120,cdcsdk_enable_dynamic_table_support=true
```

- **Positioning by End of Log (EOL) or End of File (EOF) is not supported:** Instead of EOL/EOF, you can alter the slot position by current timestamp, which will filter the existing records.
- **Truncate Operation:** Parameters such as `GETTRUNCATES/IGNORETRUNCATES` are not supported as the database does not allow user to truncate the table if they are enabled for change data capture.
- **WAL Retention:** Yugabyte normally purges WAL segments after some period of time. This means that Oracle GoldenGate might lose the WAL log data if the logs are truncated when the Extract is down for longer duration. The WAL retention is not guaranteed in Yugabyte until a couple of parameters such as `cdc_intent_retention_ms` and `cdc_wal_retention_time_secs` are adjusted to retain the WAL logs for longer duration. This should be adjusted in all the nodes or cluster where the Oracle GoldenGate Extract process is running.

The recommended values are as follows:

```
cdc_wal_retention_time_secs=86400
```

```
cdc_intent_retention_ms = 86400000
```

- **UNREGISTER EXTRACT command:** Oracle GoldenGate uses the `UNREGISTER EXTRACT` command to drop the replication slot. A replication slot can only be dropped after the client consuming from it has been stopped for at least `ysql_cdc_active_replication_slot_window_ms` duration. The default value of the flag is 5 minutes, which means you must wait for the maximum time window for which a replication slot can remain inactive before being considered stale.

In case of a cluster setup, this parameter needs to be adjusted in all the nodes with the same value.

- **BLOB/CLOB limitations:** The data type such as text/image has the limitation to capture only 200MB of data. The row size should not exceed more the 32MB and each column containing the lob data should be kept up to 2MB.

<https://docs.yugabyte.com/stable/develop/best-practices-ysql/#row-size-limit>

- **Limitations with SAVEPOINT and Rollback in DML Operations:** Oracle GoldenGate CDC Extract does not support the `SAVEPOINT` rollback operations. The CDC test_decoding plugin does not honor `ROLLBACK TO SAVEPOINT` commands. As a result, records that should have been excluded due to a rollback are still captured and processed by the Extract.

This behavior is a known limitation of the Yugabyte CDC SDK module. To know more, refer to the following links:

<https://yugabyte.atlassian.net/browse/DB-11714>

<https://docs.yugabyte.com/stable/develop/change-data-capture/using-logical-replication/#limitations>

- **Drop Table limitation:** Once a slot is created, user must not drop any table from the database until the slot is dropped. If the slot is active and user tries to drop any table from the database, then CDC Extract will not be able to capture any records from other CDC tables which are part of replication. Also, it is suggested that when CDC extract is running then also user should not drop any table on the database where the slot is created. If do not follow this, then CDC Extract will abend with the disconnection error similar to the following:

Error: The server has ended the streaming connection.

- **Limitation on statement_timeout:** Yugabyte parameter statement_timeout should not be modified at the user or the database level, and should be kept to default. If it is set to lower value, then the CDC Extract will not be able to read the records from the WAL. If required, it should always be set at the session level only.
- **Creation of slot in the replica node with read only mode is not supported:** The CDC Extract will not able to capture the records from the replica node. This is because the version that we support like 2024.2.1.0 or 2024.2.2.0 are build on top of PostgreSQL 11.2, which does not support creating the slot in the replica node.

TimesTen

With Oracle GoldenGate for Oracle TimesTen, you can deliver initial load and transactional data from other supported Oracle GoldenGate sources, such as an Oracle database.

Oracle GoldenGate for Oracle TimesTen supports data filtering, mapping, and transformations unless noted otherwise in this documentation.

Prepare Database Users and Privileges for TimesTen

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

- Oracle GoldenGate processes require a database user to deliver data to a TimesTen database and it is recommended to create a dedicated TimesTen database user for Replicat:
 - Replicat (target database)
 - DEFGEN (target database)
- 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, the Oracle GoldenGate database user.
- For Oracle GoldenGate to replicate to a target Oracle TimesTen database, grant CREATE SESSION, SELECT, INSERT, UPDATE, and DELETE on all the target tables to the Replicat database user.
- For creating heartbeat and checkpoint tables, grant CREATE ANY VIEW to the Replicat database user.

Prepare Database Connection

Learn about configuring database connections for Oracle GoldenGate for TimesTen.

Choosing the Oracle TimesTen Connectivity Type

Oracle TimesTen supports two distinct connectivity types for tools, utilities and applications; Direct mode and Client-Server mode.

Oracle GoldenGate supports both types of connectivity for Oracle TimesTen, so you can use whichever is most appropriate, based on your requirements.

Direct mode

Direct mode is a highly optimized local connectivity mechanism that eliminates interprocess communication (IPC) between the application and the database. It provides low latency and high throughput with low overhead. With Direct mode, the client application and the database must reside on the same host.

Client-Server mode

Client-Server mode is a traditional TCP/IP based connection mechanism. In this mode, the client application may reside on the same host as the database but more commonly it is installed on a different host and connects over the network. Client-Server mode has lower performance than Direct mode due to additional overhead and network latency.

Configure a DSN Connection in Linux

Oracle GoldenGate for TimesTen connects to TimesTen using the ODBC API (TimesTen's native API). ODBC connectivity defines the concept of a Data Source Name (DSN). A DSN is a logical name which applications use to specify the parameters to be used for connecting to a target database.

When using the Direct mode connectivity, connections must reference a *server* DSN defined in the `sys.odbdc.ini` file of the Oracle TimesTen instance that hosts the database (the server instance). The `sys.odbdc.ini` file is located in the `tt_instance_home/conf` directory.

When using the Client-Server mode, connections must reference a *client* DSN defined in the `sys.odbdc.ini` file of either the Oracle TimesTen instance that manages the database (the server instance) or, more commonly, in the `sys.odbdc.ini` of an Oracle TimesTen client instance, such as an Oracle GoldenGate hub server.

For more information on defining Oracle TimesTen server and client DSNs, refer to [Oracle® TimesTen In-Memory Database - Operations Guide](#).

The following is an example of a `sys.odbdc.ini` entry to define a client DSN called `ttrmtc`, that connects to database server `tthost.mydomain.com` with port 6625, and server DSN `myttddb`.

```
[ODBC Data Sources]
ttrmtc=TimesTen 22.1 Client Driver
[ttrmtc]
TTC_SERVER=tthost.mydomain.com/6625
TTC_SERVER_DSN=myttddb
```

Note

You can also set the ODBCINI environment variable with the location of the `odbc.ini` file during the Oracle GoldenGate Microservices Architecture installation. However, the `sys.odbc.ini` environment variable takes precedence over `odbc.ini`. If the DSN is not found in the `sys.odbc.ini` file then the `odbc.ini` file is searched for the DSN value.

To set up the database connection from Oracle GoldenGate for a TimesTen deployment after the DSN entry has been created, see [Add Database Connections](#).

Preparing Tables for Processing

The following table attributes must be addressed in an Oracle GoldenGate environment for TimesTen.

Removing ON DELETE CASCADE Constraints

If a target table in Oracle TimesTen has a foreign key, which specifies the `ON DELETE CASCADE` clause, and if the table that is the target of that foreign key is also a target for Oracle GoldenGate replication then you must remove the `ON DELETE CASCADE` clause from the foreign key definition to avoid errors.

Consider the following example, where the source tables are `emp_src` and `salary_src` and the target tables are `emp_targ` and `salary_targ`.

1. A delete is issued for `emp_src`.
2. It cascades a delete to `salary_src`.
3. Oracle GoldenGate sends both deletes to the target.
4. The parent delete arrives first and is applied to `emp_targ`.
5. The parent delete cascades a delete to `salary_targ`.
6. The cascaded delete from `salary_src` is applied to `salary_targ`.
7. The row cannot be located because it was already deleted in step 5.

Oracle TimesTen does not support the disabling of foreign key constraints or on delete cascade constraints. To remove the `ON DELETE CASCADE`, you must either drop the table and recreate it without the `ON DELETE CASCADE` clause or you can use `ALTER TABLE` to remove the foreign key constraint and recreate it without the `ON DELETE CASCADE` clause.

Ensuring Row Uniqueness for Tables

Oracle GoldenGate requires some form of 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:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.

3. 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 those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration.

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.

4. If a table does not have an appropriate key, or if you prefer that the existing key(s) are not 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. See TABLE | MAP in *Parameters and Functions Reference for Oracle GoldenGate*.

TimesTen: Supported Data Types, Objects, and Operations

This section contains information on supported data types, objects, and operations for Oracle GoldenGate on TimesTen.

Supported TimesTen Data Types

The following data types are supported for delivery, unless specifically noted in the limitations that follow:

- Binary Data Types
 - binary
 - varbinary
- Character Data Types
 - char
 - nchar
 - nvarchar2
 - varchar2
- Date and Time Data Types
 - date
 - time
 - timestamp
 - tt_date
 - tt_time
 - tt_timestamp
- Numeric Data Types

- binary_float
- binary_double
- double
- float
- tt_bigint
- tt_integer
- number
- real
- tt_smallint
- tt_tinyint
- LOB
 - blob
 - clob
 - nclob

The binary_float and binary_double data types support the NaN and Infinity/-Infinity special numeric values.

For more information on data types by databases that support special values, refer to [Special Values Supported by Database Data Types](#).

Supported Objects and Operations for TimesTen

The following objects and operations are supported:

- Oracle GoldenGate for Oracle TimesTen supports delivery of transactional DML to user tables.
- INSERT, UPDATE, and DELETE operations are supported.

Non-Supported TimesTen Data Types and Features

The INTERVAL and ROWID data types are not supported.

Limitations and Non-supported Items for Oracle TimesTen

The limitations and non-supported items for Oracle TimesTen are:

- Capture (extraction) of DML operations is not supported.
- Capture and replication of DDL (data definition language) operations is not supported.
- The support of 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.
- 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 time zone, conversion may add or subtract hours, which can cause the timestamp to exceed the lower or upper supported limit.

- Modifying the Primary Key Column value is not supported.
- Limitations on Automatic Heartbeat Table support are as follows:
 - Oracle GoldenGate supports only Delivery on TimesTen so no mechanism is required to populate/update the heartbeat tables using any event/job schedulers.
 - The ALTER HEARTBEATTABLE command is not supported and if used is ignored.
 - The ADD HEARTBEATTABLE command with the FREQUENCY, PURGE_FREQUENCY, or RETENTION_TIME option is not supported. When any of these options are specified with the ADD HEARTBEATTABLE command, a warning is displayed that the option is ignored.
 - Since TimesTen does not have any internal event/job schedulers, automatic purging of heartbeat history tables *cannot* occur. As such, you should explicitly drop or truncate the corresponding heartbeat objects to suit your environment.

Prepare Oracle GoldenGate

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

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
Userid: 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](#)

Authenticate the Database Connection Using the Database Password Plugin

Oracle GoldenGate 23ai (23.5 and higher) includes a user-friendly plugin that enables writing custom code to retrieve database credentials from a third-party vault and then return it in a predefined API to Oracle GoldenGate.

With increased security rules included as best practices by organizations, some organizations do not allow storing the database credentials, such as password, in Oracle GoldenGate's local wallet (credential store). The password needs to be stored in a vault outside of Oracle GoldenGate. To implement this method of authentication, the Database Password Plugin has been introduced. It allows Oracle GoldenGate to retrieve the password from any third-party vault of your choice or the OCI Vault, when it needs to connect to a database instance.

Learn about the workflow to enable the Database Password Plugin in Oracle GoldenGate and use it to retrieve the secret password, when establishing a database connection from Oracle GoldenGate.

Use any of the following options to setup a database password plugin service for Oracle GoldenGate:

- [Configure the Database Password Plugin to Store and Access the Database Password from a Third-Party Vault](#)
- [Configure the Database Password Plugin to Store and Access the Database Password from OCI Vault](#)
- [Configure the Database Password Plugin to Store and Access the Database Password from Oracle Key Vault](#)

APIs Required for Database Password Plugin

A minimum of two function APIs need to exist in this plugin. These APIs are described as follows:

getManifest()

Each plugin must expose the `getmanifest` function that returns a JSON style string describing the plugin, as shown:

```
{"$schema": "ogg:plugin", "version": "23.4" }
```

The version number can be equal to or lower than the Oracle GoldenGate version number. For example, if you are running Oracle GoldenGate 23ai (version 23.6), then the version number in the JSON string could be 23.4, 23.5, or 23.6.

getUseridAliasPassword(oggDomain *C.char, oggAlias *C.char, oggUserId *C.char, oggPassword *C.char)

This function is called by Oracle GoldenGate to retrieve the password for a `USERIDALIAS`.

Returns an integer value. 0 means success and any non-zero value is a failure.

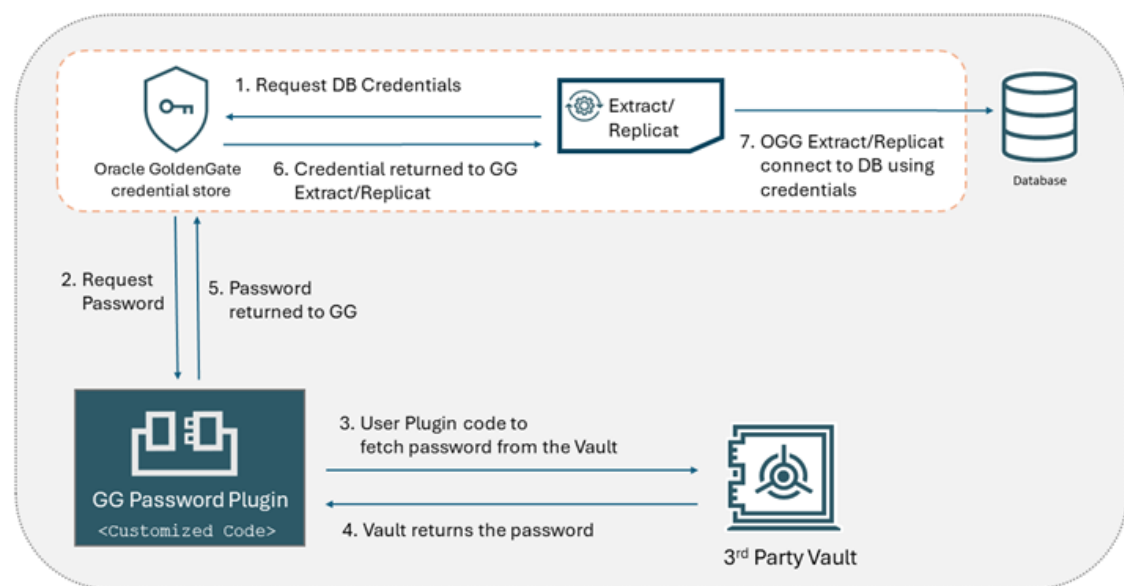
`oggDomain`, `oggAlias`, and `oggUserId` are the input parameters to identify the database user credentials. `oggPassword` is the output parameter that returns the password fetched from the vault.

Configure the Database Password Plugin to Store and Access the Database Password from a Third-Party Vault

To create, store, and retrieve the secret password from a third-party vault of your choice, you need to write your own custom code. The code needs to generate a shared user library, which would be used to activate the plugin as a service from Oracle GoldenGate Service Manager. After the plugin is activated, you can access the third-party vault to retrieve the stored secret password, when establishing a database connection from Oracle GoldenGate. Oracle GoldenGate stores that *USERID* as an alias in its credential store under *domain_name.alias_name*. The *domain_name* and *alias_name* are used as input parameters to call the plugin. For example, if the domain name is **OracleGoldenGate** and the alias is **ggestadmin**, then the secret name would be **OracleGoldenGate.ggestadmin**. Plugin identifies the secret name to retrieve the secret (password) from the vault.

The custom code can be written in any language as long as it can be compiled into a shared user library. The most popular languages that can be used include GoLang, C/C++, and Python.

The following diagram shows the workflow when the Database Password Plugin is used with Oracle GoldenGate to set up a database connection from a third-party vault. The *USERID* ALIAS is stored in Oracle GoldenGate credential store and the secret password is stored in the third-party vault.



Use the following steps to configure the Database Password Plugin for storing the Database password in a third-party vault and retrieving it to authenticate a database connection from Oracle GoldenGate:

1. Locate the Database Password Plugin, which is present in the **GG Service Manager Home/var/lib/plugins/** directory. By default, there is no library under this directory.

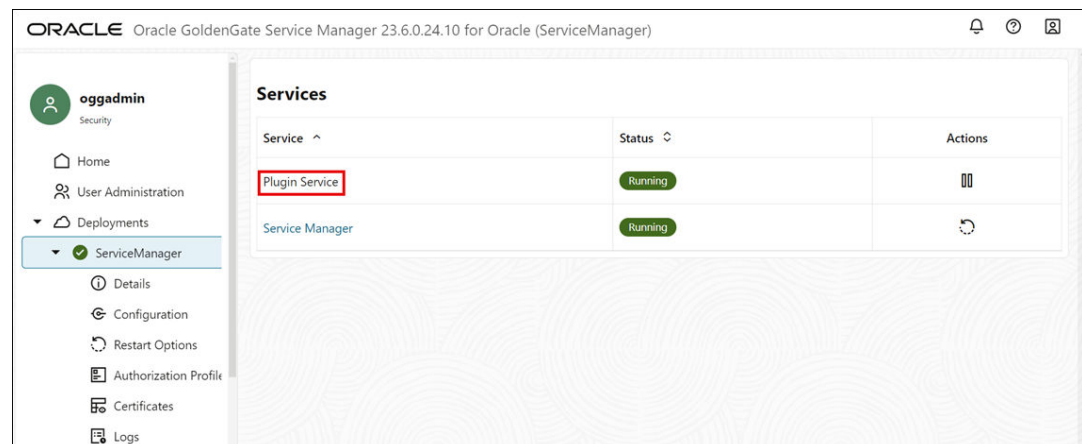
Note

Although there are no restrictions on the library file name, the plugin will pick up only the first library file that exists in this location.

2. Check the function APIs used with the plugin. See [APIs Required for Database Password Plugin](#) for details.
3. Write and compile the code to create a shared user library that would be used to activate the Database Password Plugin. You need to write and compile the code, based on the third-party vault that you are using to set up the Database Password Plugin.
4. Copy the compiled user library to the Service_Manager_Home/var/lib/plugins/ directory.
5. Run the REST API command to activate the plugin as a new service under Service Manager. Assuming that the Service Manager is running locally on port 9011, then the command will look similar to the following:

```
shell>OGG_ADMIN_PASSWORD="xxxx"
shell>curl -svu "oggadmin:${OGG_ADMIN_PASSWORD}"
http://127.0.0.1:9011/services/v2/deployments/ServiceManager/services/
pluginsrvr \ -XPOST --data '{"$schema":"ogg:service", "config":"external",
"enabled":true,
"critical":true, "status":"running"}'
```

You can view the status of the plugin from the Service Manager after it is activated, as shown in the following image. You can view the service as **Plugin Service** under **Services**.

**Note**

If you are using NGINX or a secured deployment, you need to use *https* instead of *http*, and the appropriate CA certificate path in the command line.

6. Set up any environment variables for the third-party vault, if required, and restart the Service Manager.
7. Configure the third-party vault and store the password in it.

- Run the Admin Client and set up the database login credentials with the NOPASSWORD option and test that the connection to the database is successful. The following example illustrates this configuration:

```
OGG (not connected) 1> CONNECT http://127.0.0.1:9011 AS oggadmin
Using default deployment 'Deployment'
```

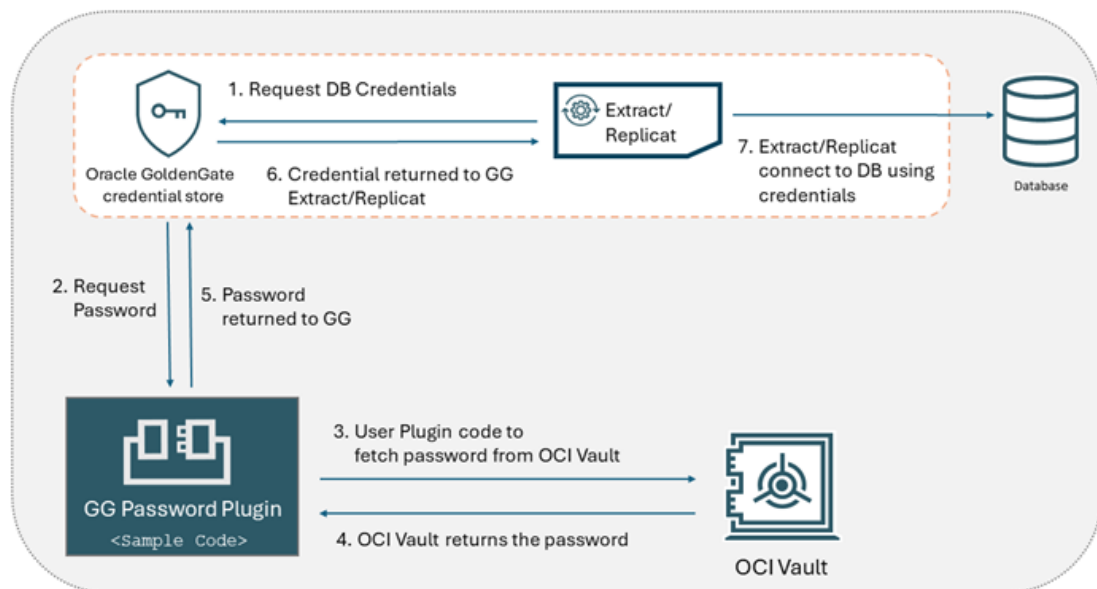
```
OGG (http://127.0.0.1:9011 Deployment) 2> ALTER CREDENTIALSTORE ADD USER
ggeastadmin@//dbms:1521/ORCLPDB1 ALIAS ggeastadmin NOPASSWORD
2024-04-02T18:34:14Z INFO OGG-15114 Credential store altered.
```

```
OGG (http://127.0.0.1:9011 Deployment) 3> DBLOGIN USERIDALIAS ggadmin
Successfully logged into database.
```

Configure the Database Password Plugin to Store and Access the Database Password from OCI Vault

To create, store, and retrieve the secret password from OCI Vault, you can use the sample code, which shows how to retrieve the secret password from the OCI Vault. Oracle GoldenGate stores the *USERID* as an alias in its credential store under *domain_name.alias_name*. The password is stored in OCI Vault as a secret under the secret name *domain_name.alias_name*. For example, if the domain name is **OracleGoldenGate** and the alias is **ggeastadmin**, then the secret name would be **OracleGoldenGate.ggeastadmin**. Plugin identifies the secret name to retrieve the secret (password) from the vault.

The steps in the diagram demonstrate the workflow for setting up the Database Password Plugin with OCI Vault:



- Create a password in OCI vault.

2. Verify that the secret is created in the OCI Vault that holds the password for this database user ggeastadmin. Make sure the secret name matches the expected naming convention, for example, **OracleGoldenGate.ggeastadmin** in the format *domain_name.alias_name*.

The screenshot displays the Oracle Cloud console interface for a Vault named GGETESTVault1. The 'General information' tab is active, showing details such as the compartment (pasdevagg [root]/GG_ProductManagement/werhe), the OCID (highlighted with a red box and labeled 'Vault ocid'), and the creation date. Below this, the 'Secrets in werhe Compartment' table is visible, listing several secrets. The first secret, 'OracleGoldenGate.ggeastadmin', is highlighted with a red box and labeled 'Secret/Password Name'.

Name	Status	Created	Auto generation	Auto rotation
OracleGoldenGate.ggeastadmin	Active	Mon, Aug 26, 2024, 05:35:18 UTC	Off	Off
OracleGoldenGate.MyATPPV	Active	Mon, Aug 26, 2024, 03:18:21 UTC	Off	Off
OracleGoldenGate.MyATP	Pending Deletion	Mon, Aug 26, 2024, 03:12:21 UTC	Off	Off
GGADMINPASSWORD	Active	Mon, Oct 9, 2023, 15:35:16 UTC	Off	Off

3. Locate the Database Password Plugin, which is present in the **GG Service Manager Home/var/lib/plugins/** directory. By default, there is no library under this directory.

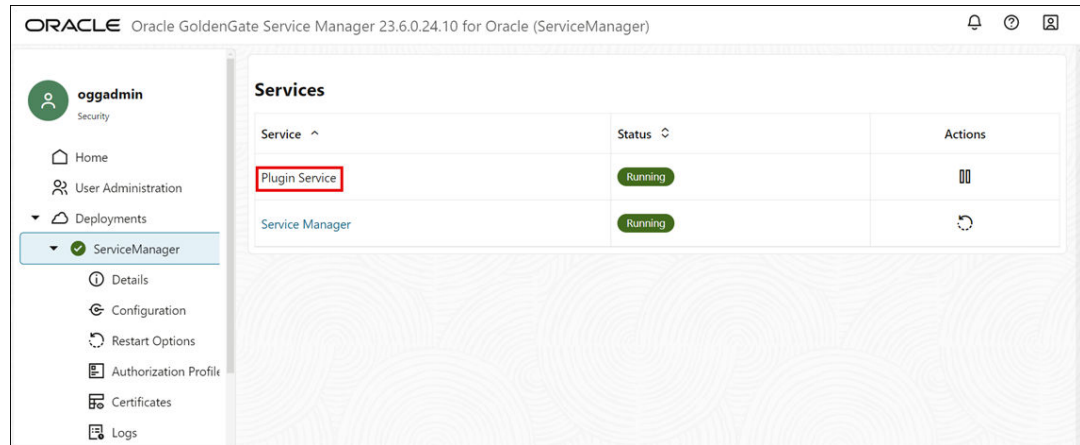
Note

Although there are no restrictions on the library file name, the plugin will pick up only the first library file that exists in this location.

4. Check the function APIs that you need to use with the plugin. See [APIs Required for Database Password Plugin](#) for details.
5. Access the sample code from OGG_HOME/src/PluginExamples/OCIVault/OCIVault.go, to generate the shared user library.
6. Compile the code by running the make command from OGG_HOME/src/PluginExamples/OCIVault/OCIVault.go. This generates the user library, named libOCIVault.so.
7. Copy the user library libOCIVault.so to the Service_Manager_Home/var/lib/plugins/ directory.
8. Issue the REST API command to activate this plugin as a new service under Service Manager. Assuming that the Service Manager is running locally on port 9011, then the command will look similar to the following:

```
shell>OGG_ADMIN_PASSWORD="xxxx"
shell>curl -svu "oggadmin:${OGG_ADMIN_PASSWORD}"
http://127.0.0.1:9011/services/v2/deployments/ServiceManager/services/
pluginsrvr \ -XPOST --data '{"$schema":"ogg:service", "config":"external",
"enabled":true,
"critical":true, "status":"running"}'
```

You can view the status of the plugin from the Service Manager after it is activated, as shown in the following image. You can view the service as **Plugin Service** under **Services**.

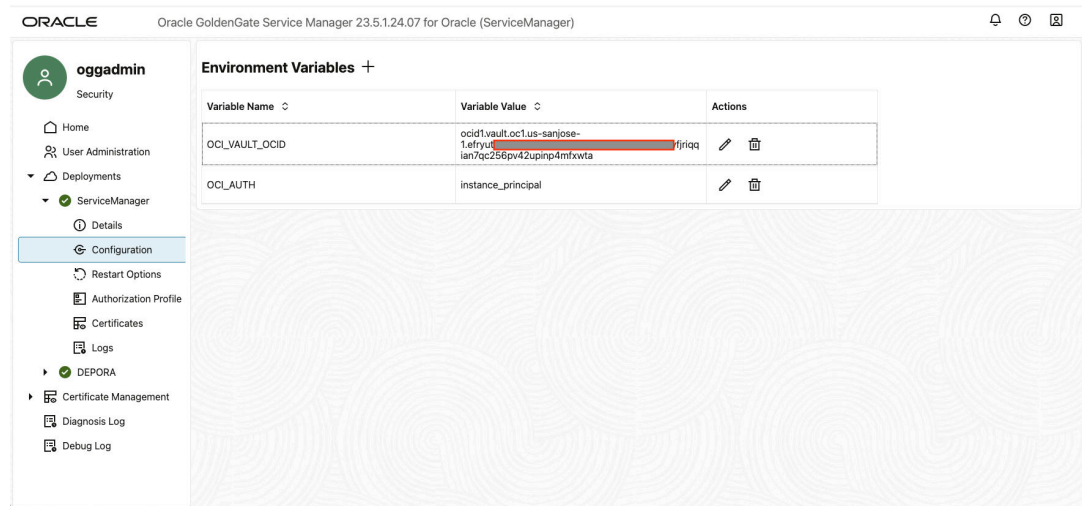


Note

If you are using NGINX or a secured deployment, you need to use *https* instead of *http*, and the appropriate CA certificate path in the command line.

9. In the Service Manager web interface, set the `OCI_AUTH` and `OCID_VAULT_OCID` environment variables to the following values:
 - **OCI_AUTH = instance_principal**: The `instance_principal` authentication method has been used in this example. However, the Database Password Plugin supports the following authentication methods for OCI Vault, including:
 - `instance_principal`: Instance Principal authentication.
 - `DEFAULT`: Default profile in the OCI configuration file.
 - `Customized name`: Customized profile in the OCI configuration file.
 - **OCID_VAULT_OCID = ocid1.vault.oc1-us-sanjose-1.exxxxxxxxxxxxxxxxxx42upinp4mfxwta**: This value is derived from step 2.

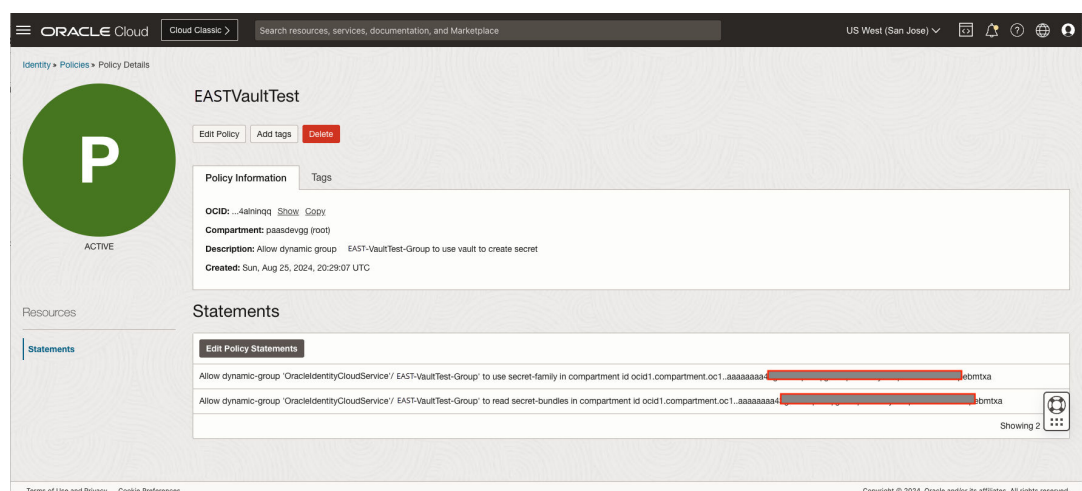
Both these environment variables must be set up correctly in the environmental variable section under the Service Manager's Configuration.



Note

If you are using NGINX or a secured deployment, you need to use *https* instead of *http*, and the appropriate CA certificate path in the command line.

10. Restart the Service Manager to activate the settings.
11. Identify the VM instance where Oracle GoldenGate Service Manager is running.
12. Add the VM instance to a dynamic group, to make `instance_principal` authentication work.
13. Assign a tag for the VM instance, for example, **EASTVaultTest**. This would make it easier to create dynamic groups. See [Managing Dynamic Groups](#) for details. The dynamic group needs a policy to use the secret in that vault.
14. Create a policy to allow this dynamic group to use the secret resource (secret password). For details on creating policy to control access to the vault service, see [Details for the Vault Service](#).



- Run the Admin Client to create Database login alias with the NOPASSWORD option and test that the connection is successful, as shown in the following example:

```
OGG (not connected) 1> CONNECT http://127.0.0.1:9011 AS oggadmin
Using default deployment 'Deployment'
```

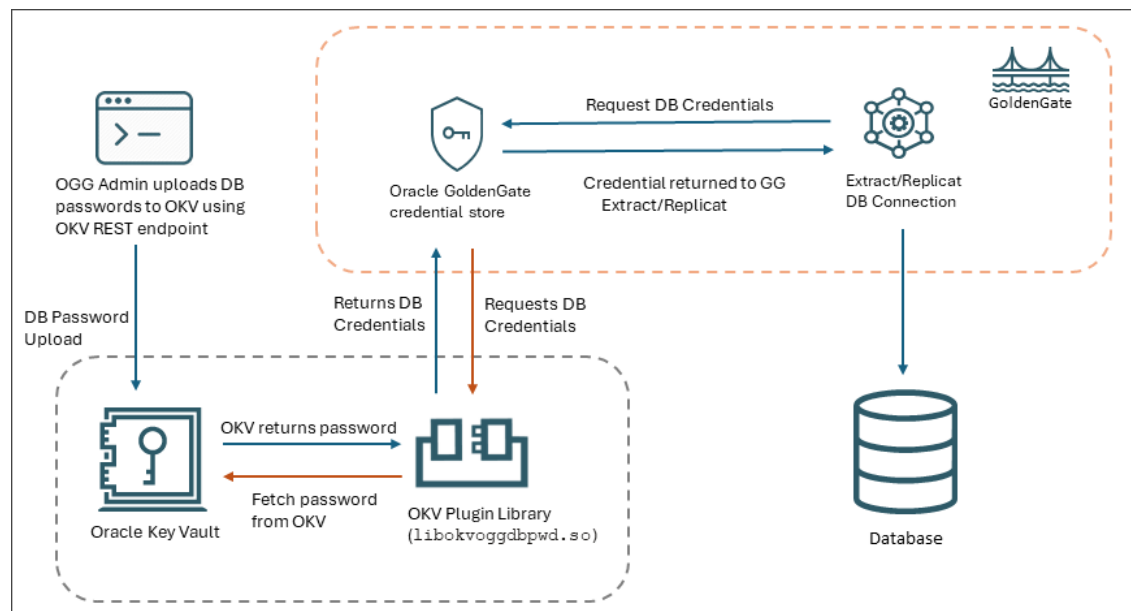
```
OGG (http://127.0.0.1:9011 Deployment) 2> ALTER CREDENTIALSTORE ADD USER
ggeastadmin@//dbms:1521/ORCLPDB1 ALIAS ggeastadmin NOPASSWORD
2024-04-02T18:34:14Z INFO OGG-15114 Credential store altered.
```

```
OGG (http://127.0.0.1:9011 Deployment) 3> DBLOGIN USERIDALIAS ggadmin
Successfully logged into database.
```

Configure the Database Password Plugin to Store and Access the Database Password from Oracle Key Vault

Oracle GoldenGate 23.5 and higher releases include a plugin service that enables you to write custom code for retrieving database credentials from a third-party vault and then return the credentials in a predefined API to Oracle GoldenGate, as mentioned in [Configure the Database Password Plugin to Store and Access the Database Password from a Third-Party Vault](#). This feature includes the `libokvoggdbpwd.so` library that can be configured in Oracle GoldenGate to manage passwords from Oracle Key Vault.

The following diagram represents the workflow between Oracle GoldenGate and Oracle Key Vault to configure the Plugin Service and then use the OKV Plugin Library to request and retrieve database passwords from Oracle Key Vault server.



The process of storing and retrieving the database passwords from Oracle Key Vault can be summarized as follows:

- Oracle GoldenGate administrator uses the RESTful services utility to upload Oracle GoldenGate passwords to Oracle Key Vault.

2. Oracle GoldenGate requests for the database credentials from Oracle Key Vault server using the `libokvoggdbpwd.so` plugin library. This password is fetched from the OKV server after successful backend validations.
3. After the password is retrieved, Oracle GoldenGate processes will be able to connect to the database using a remotely stored password.

See the Centralizing the Management of Oracle GoldenGate Passwords in the *Oracle Key Vault Administrator's Guide* for details about implementing this feature for Oracle GoldenGate database password storage and retrieval.

Add Database Connections

You must have working database connections for your Extract and Replicat processes. Follow these steps to create database connections to connect your source and target hosts for replication.

1. Launch the Administration Service interface and log in.
2. Click **DB Connections** from the left navigation pane.
3. Click the plus sign (+) sign next to **DB Connections**. The Credentials dialog box is displayed.
4. Enter the following details in the displayed fields:

Note

All the fields listed below may not be applicable to all the databases. Review the details of each field to know if it is applicable to a database.

- **Credential Domain**

Specify a domain name to which the database credential is associated. For example, "OracleGoldenGate" is the default domain name, in case you don't specify a domain name.

- **Credential Alias**

This is the alias for your database credential, such as ggeast.

- **Connection Type**

Type of connection to be used to connect to the database. The following two connection types are available:

- **Data Source Name**

The DSN entry for the database connection.

- **Database Server and Port**

On selecting this option, the **Database Hostname or IP Address** and **Port Number** fields are displayed. Specify the database hostname or IP address and port number, respectively.

- **Data Source Name**

The DSN entry for the database connection.

- **Database Server / Database Hostname or IP Address**

Name or IP address of the server where the database is running.

- **Port / Port Number**
Port number for connecting to the database server.
- **Database Name**
Name of the database to which you want to connect.
- **User ID**
User ID or username of the database user.

Note

If the user name in the database has a dash in it, for example OGG-ADMIN, then the following error occurs:

```
OGG-08221 Cannot register or unregister EXTRACT extract_name
because of the following SQL
error: OCI Error ORA (status = 44004-ORA-44004: invalid
qualified SQL name.
```

- **Password and Verify Password**
Password used by database user to log in to the database.
- **Security Protocol**
Security protocol that is used to connect to the database. You can use any of the following three security protocols:
 - a. **Plaintext**
It is the default security protocol when **Connection Type** is **Database Server and Port**.
 - b. **TLS**
On selecting this option for SQL Server, the **Trust Server Certificate** option is selected by default. If you deselect the **Trust Server Certificate** option, you would need to specify the values for **SSL Path** and **SSL Certificate** options.
 - c. **mTLS**
When **TLS** option is selected, the following two sub-options are displayed:
 - **SSL Mode**: The available options are:
 - **PREFERRED, REQUIRED, VERIFY_CA**, and **VERIFY_IDENTITY** for MySQL.
 - **PREFER, REQUIRE, VERIFY-CA, VERIFY FULL** for PostgreSQL.Select the applicable SSL mode.
On selecting **PREFERRED** or **REQUIRED** for MySQL or **PREFER** or **REQUIRE** for PostgreSQL, specify any additional applicable attributes in the **Additional Attributes** text box.
On selecting **VERIFY_CA** or **VERIFY_IDENTITY** SSL mode for MySQL or **VERIFY_CA** or **VERIFY_IDENTITY** SSL mode for PostgreSQL, the following fields are displayed:
 - **SSL Path**: The path name of the directory that contains trusted SSL Certificate Authority (CA) certificate files.
 - **SSL CA**: The path of the CA certificate file in PEM format.

- **SSL CRL:** The path of the file containing certificate revocation lists in PEM format.
- **Additional Attributes:** Specify any additional information in this field.

When **mTLS** mode is selected, the following sub-options are displayed:

- **SSL Mode:** The available options are:
 - **PREFERRED, REQUIRED, VERIFY_CA,** and **VERIFY_IDENTITY** for MySQL.
 - **PREFER, REQUIRE, VERIFY-CA, VERIFY FULL** for PostgreSQL.

On selecting **PREFERRED** or **REQUIRED** for MySQL or **PREFER** or **REQUIRE** for PostgreSQL, the following fields are displayed:

- **SSL Path:** The path of the directory that contains trusted SSL Certificate Authority (CA) certificate files in PEM format.
- **SSL Certificate:** The path of the SSL public key certificate file in PEM format. On the client side, this is the client public key certificate and on the server side, this is the server public key certificate.
- **SSL Key:** The path of the SSL private key file in PEM format. On the client side, this is the client private key, and on the server side, this is the server private key.

On selecting **VERIFY_CA** or **VERIFY_IDENTITY** SSL mode for MySQL or **VERIFY_CA** or **VERIFY_IDENTITY** SSL mode for PostgreSQL, the following fields are displayed, in addition to the fields listed above:

- **SSL CA:** The path of the CA certificate file in PEM format.
- **SSL CRL:** The path of the file containing certificate revocation lists in PEM format.
- **Additional Attributes:** Specify any additional information in this field.

5. Click **Submit**.
6. Click the **Connect database** icon to verify that the connection is working correctly. If the connection is successful, the Connect to database icon turns blue. You'll also see sections to set up checkpoint and heartbeat tables after the connection is successful.
7. From the DB Connections page, you can also perform the following tasks from the Action column:
 - Connect to database
 - Copy DB Connection
 - Alter the DB Connection
 - Delete the DB Connection

After successfully connecting to the database, you can add TRANDATA, SCHEMATRANDATA, checkpoint, and heartbeat tables required by Extract and Replicat.

Add a Path Connection

A path connection is required when using a Distribution Path for connecting between two deployments on remote servers. In a hub-based environment you may not need to set up a Distribution Path for connecting the deployments.

Use the Path Connection page to create user credentials for the distribution path to connect to two separate deployments across a secure network. Follow these steps to configure a path connection from the Administration Service:

Note

Before you begin adding a distribution path, make sure you establish a path connection. See [Add a Path Connection](#).

1. From the Administration Service left navigation pane, click Path Connections.
2. From the Path Connections page, click the plus sign to open the Path Connection dialog box.
3. Enter the following options in the Path Connection dialog box:
 - **Credential Domain:** This option is preset as Network and cannot be changed.
 - **Credential Alias:** Specify an alias for the user credential, such as ggeast.
 - **User ID:** Specify a user ID in this box. For a distribution path connection, this value is the user ID used to access the target deployment.
 - **Password:** Specify a password for the associated user ID.
4. Click **Submit**.

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.

Enable TRANDATA for Non-Oracle Databases

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

This can be done at the table 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** option 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, such as, `EMPLOYEE`. You can also use wildcard instead of specific table name.
3. Click **Submit**.

To Enable Change Capture for Db2 for i

To capture changes to a table in a journal, you can run the `STRJRNPF` command on the IBM i command line or run the `ADD TRANDATA` command from Admin Client. The `ADD TRANDATA` command calls `STRJRNPF` and is the recommended method to start journaling for tables, because it ensures that the required journal image attribute of Record Images (`IMAGES`): `*BOTH` is set on the `STRJRNPF` command.

The `ADD TRANDATA` command is:

`ADD TRANDATA table_specification`

Where: `table_specification` is one of the following:

- `schema.table [JOURNAL library/journal]`
- `library/file [JOURNAL library/journal]`

Specifying a Default Journal

To specify a default journal for multiple tables or files in the `ADD TRANDATA` command, instead of specifying the `JOURNAL` keyword, use the following command before issuing `ADD TRANDATA`.

`DEFAULTJOURNAL library/journal`

Any `ADD TRANDATA` command used without a journal assumes the journal from `DEFAULTJOURNAL`.

To display the current setting of `DEFAULTJOURNAL`, you can issue the command with no arguments.

Removing a Default Journal Specification

To remove the use of a default journal, use the following command:

```
DEFAULTJOURNAL CLEAR
```

To Enable Change Capture for Db2 LUW

Configure Db2 to log data changes in the expanded format that is supplied by the `DATA CAPTURE CHANGES` feature of the `CREATE TABLE` and `ALTER TABLE` commands. This format provides Oracle GoldenGate with the entire before and after images of rows that are changed by update statements. Run the `ALTER TABLE` command as follows:

Issue the following command.

```
ADD TRANDATA owner.table
```

where *owner.table* is the fully qualified name of the table. You can use a wildcard to specify multiple table names.

`ADD TRANDATA` issues the following command, which includes logging the before image of `LONGVAR` columns:

```
ALTER TABLE name DATA CAPTURE CHANGES INCLUDE LONGVAR COLUMNS;
```

Example 7-1 To Exclude LONGVAR Logging:

To omit the `INCLUDE LONGVAR COLUMNS` clause from the `ALTER TABLE` command, use `ADD TRANDATA` with the `EXCLUDELONG` option.

```
ADD TRANDATA owner.table, EXCLUDELONG
```

Note

If `LONGVAR` columns are excluded from logging, the Oracle GoldenGate features that require before images, such as the `GETUPDATEBEFORES`, `NOCOMPRESSUPDATES`, and `NOCOMPRESSDELETES` parameters, might return errors if tables contain those columns. For a workaround, see the `REQUIRELONGDATAACAPTURECHANGES` | `NOREQUIRELONGDATAACAPTURECHANGES` options of the `TRANLOGOPTIONS` parameter.

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. A checkpoint either succeeds or fails with the transaction, so Replicat ensures that a transaction is only applied once, even if there is a failure of the process or the database.

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 a Checkpoint Table with Journaling in Db2 for i

In the Db2 for i target database, each checkpoint that is written to the checkpoint table, must be journaled within the Replicat transaction.

A common method to create the checkpoint table with journaling is as follows:

1. On the target system, create the Replicat checkpoint file.

Set the name of the journal that is intended to be used for the checkpoint file. The default will be `GGSCHEMA/OGGJRN`.

You can change it by setting the default for the current Admin Client session using the `DEFAULT JOURNAL` command. The syntax of the `DEFAULT JOURNAL` command is:

```
DEFAULT JOURNAL library_name/journal_name
```

Where: `library_name` is the name of the library and `journal_name` is the name of the journal to be used for subsequent operations that may optionally have a journal specified.

2. Add the checkpoint table.

```
ADD CHECKPOINTTABLE library_name.chkptab
```

3. Add journaling to the checkpoint table.

```
ADD TRANDATA library_name.CHKPTAB
```

For more information about creating a checkpoint table, see [Add a Checkpoint Table](#).

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 `ggeist`:

```
(https://remotehost:16000 ggdep_postgres)> DBLOGIN USERIDALIAS ggeist
```

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.

Running the Heartbeat Update and Purge Function for PostgreSQL

Oracle GoldenGate for PostgreSQL supports a heartbeat table configuration, with some limitations regarding the update and purge tasks.

The heartbeat table and associated functions are created from the `ADD HEARTBEATTABLE` command, however for PostgreSQL, there is no automatic scheduler to call the functions.

One main function controls both the heartbeat record update and the heartbeat history table purge functions. The default settings for both of these features are 60 seconds for the update frequency and 1 day for the history record purge, which deletes all records older than 30 days by default.

To call the main heartbeat record function, users should create an operating system level job that executes

```
"select ggschema.gg_hb_job_run();" 
```

When this function is called, it will take into account the update frequency settings and history record purge settings and use those values regardless of the scheduler settings for the function call.

For example, users can create a **Cron Job** with the following syntax, and have it run every minute.

```
*****PGPASSWORD="gguserpasswd" psql -U gguser -d dbname -h remotehost -p 5432  
-c "select ggschema.gg_hb_job_run();" >/dev/null  
2>&1
```

Windows Task Scheduler, **pgAdmin**, or **pg_cron** are other programs that could be used to schedule the function call.

Create the Oracle GoldenGate CDC Cleanup Task

For SQL Server users, there is a requirement to create Oracle GoldenGate CDC Cleanup tasks before adding an Extract. You can do so by performing the steps in [Details of the Oracle GoldenGate CDC Cleanup Process](#).

PostgreSQL: Extract Considerations for Remote Deployment

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.

Prepare Oracle GoldenGate for Db2 for i

Learn how to prepare Oracle GoldenGate to begin data replication from a source Db2 for i database and apply it to a supported target database.

Update the GLOBALS File

The GLOBALS parameter file contains parameters that affect all processes within an Oracle GoldenGate instance.

GGSCHEMA is a mandatory parameter for Oracle GoldenGate and defines the schema, which Oracle GoldenGate uses on the remote system for necessary Oracle GoldenGate database objects.

The GLOBALS parameter NAMECCSID is specific to Db2 for i and may be required, if the SQL catalog contains object names that are referenced by a different CCSID than the system CCSID. The SQL catalog is created in the system CCSID and does not indicate this difference when queried. Oracle GoldenGate makes queries to the catalog and could retrieve the name incorrectly unless NAMECCSID is used to supply the correct CCSID value. For more information, see *Parameters and Functions Reference for Oracle GoldenGate*.

Specifying Object Names

Oracle GoldenGate commands and parameters support input in the form of SQL names, native names in the format of *library_name/file_name(member_name)*, or a mix of the two. If a native file system name does not include the member name, all members are implicitly selected by the Oracle GoldenGate process. For a SQL name only the first member is used.

To support case sensitivity of double quoted object names, specify those names within double quotes in the Oracle GoldenGate parameter files. This is true of both SQL and native file system names.

When specifying a native table name in a MAP statement on a platform other than DB2 for i, the name must be enclosed within double quotes so that Oracle GoldenGate correctly interprets it as a separator character.

For consistency of terminology in other administrative and reference Oracle GoldenGate documentation, the SQL terms "schema" and "table" are used to reference the containers for the DB2 for i data, as shown here.

Table 7-7 Native-SQL object name relationships

Native	SQL	Notes
Library (maximum length 10)	Schema (maximum length 128)	The operating system creates a corresponding native name for a SQL-created schema.
File (maximum length 10)	Table (maximum length 128)	The operating system creates a corresponding native name for a SQL-created table.
Member	Not Applicable	Contains the actual data. Only the first member of a FILE object can be accessed through SQL. To access data in other members the native system name must be used.

Adjusting the System Clock

It is recommended that you set the system clock to UTC (Universal Time Coordinate) time and use the timezone offset in the DB2 for i system values to represent the correct local time. If this

setup is done correctly, local daylight savings time adjustments can occur automatically with no disruption to replication.

8

Replicate Data

Learn about the complete data replication cycle including capturing trail data using Extract, instantiating data replication using different instantiation methods for different databases, distributing data across remote networks and deployments, and replicating data using the Replicat processes.

Extract

Learn about different types of Extract and how to add and manage Extracts.

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 [Precise Instantiation for Oracle Using Initial Load Extract](#).
- **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, Amazon RDS, and Google Cloud SQL databases.

Add an Online Extract

Before you begin adding an Extract, make sure that the following settings are configured:

- [Add Database Connections](#)
- [Add TRANDATA](#)
- [Add a Checkpoint Table](#)
- [Add Heartbeat Table](#)
- [Register an Extract from the Admin Client](#) for Oracle and PostgreSQL

Now, you're ready to add an Extract for your deployment.

Extract Information Screen

After logging into the Microservices home page, click the + sign in the Create column to open the **Add Extracts** wizard.

On the **Extract Information** screen, select the type of Extract. Depending on the database platform, the types of Extracts that are available may be **Integrated Extract**, **Change Data Capture Extract**, and **Initial Load Extract**.

If you need to set up the **Downstream Extract** for Oracle database, enable the **Downstream Extract** toggle switch.

Select the type of Extract to create, and specify the following:

- **Process Name:** Name of the Extract process. The name of the Extract process can be up to 8 characters.
- **Description:** Description of the Extract process being created

Note

To learn about creating initial load Extract, see [About Instantiation](#).

Click **Next**.

Register Extract Screen (PostgreSQL)

Note

Both Oracle and PostgreSQL Extracts need to be registered. However, the configuration options to register the Extract are displayed on different screens for Oracle and PostgreSQL.

Note

Also see the [Guidelines and Limitations for PGOUTPUT Plugin Type](#).

On the **Register Extract** screen, configure the following options and click **Register**:

- **Source Credentials:** Specify a domain for the database.
- **Alias:** Specify the user ID alias used as the database connection for the source login or select from the displayed options.
- (PostgreSQL only) **Plugin Type:** Select the type of logical decoding plugin: **TEST_DECODING** or **PGOUTPUT**, as shown in the following diagram.

Add Extract

Extract Information Register Extract Extract Options Managed Options Parameter File

Create a replication slot to register the extract.

Source Credentials

Domain: OracleGoldenGate

Alias: edb

Output Plugin Type (Plugin Type cannot be changed after registration)

Plugin Type: PGOUTPUT

TEST_DECODING

PGOUTPUT

Back Register

If you don't select a plugin type then the default plugin type, **TEST_DECODING**, is selected.

Also see [Register an Extract from the Admin Client](#). Click **Next** after the Extract registration is successful.

Add Extract

X

✓

3

4

5

Extract Information

Register Extract

Extract Options

Managed Options

Parameter File

Create a replication slot to register the extract.

Source Credentials

Domain
OracleGoldenGate

▼

Alias
edb

▼

Output Plugin Type (Plugin Type cannot be changed after registration)

Plugin Type
PGOUTPUT

▼

❏ Extract registered successfully

02:37 PM

A replication slot has been created for extract EXTN in the database

◀ Back

Next ▶

Click **Next**.

Add the Extract Group

On the **Extract Options** screen configure the following settings:

- **Extract Trail:**
 - **Name:** Name of the Extract trail file. The name of the trail file can be up to 2 characters.
 - **Subdirectory:** Directory name of the subdirectory where the Extract trail is stored.
 - **Trail Sequence:** Sequence number of the trail.
 - **Trail Size:** Maximum size of the trail file.
 - **Encryption Profile:** Description of the encryption profile. If you have not created an encryption profile, then the Local Wallet profile would be selected, by default. To know more about creating and applying encryption profiles, see [Configure an Encryption Profile](#).
 - **Encryption Algorithm:** List of encryption algorithms available for the Extract trail file.

Note

For more information on trail file encryption, see [Trail File Encryption](#) and [Encrypting Trail Files](#).

- (Oracle only)**Registration Options**: Specify the following options to register an Extract for Oracle database.
CSN: Commit Sequence Number (CSN) value.
- **Share**: This drop down is used to define how to share the LogMiner data dictionary. The available options are Automatic, None, and Extract Name. **Automatic** means that the system decides which Extract to share. **None** means that the LogMiner data dictionary is not shared. **Extract Name** means that the LogMiner data dictionary is shared with the specified Extract.
- **Optimized**: Enable this option to optimize the Extract registration.

Downstream Capture Screen

Valid for Oracle only.

If you selected the **Downstream Capture** on the **Extract Information** screen, then the **Downstream Capture** options screen is displayed. Configure the downstream mining database connection for the downstream Extract using this screen.

- **Mining Credentials**: Specify the domain and the user ID alias value in the **Domain** and **Alias** boxes.
- **No UserID/No Source DB Connection**: Enable this toggle switch to set up the mining database connection using Active Data Guard (ADG). The options to enter the **ADG Fetch Credential** are displayed:
 - **Domain**: Domain name for the ADG fetch database.
 - **Alias**: Domain alias for the ADG fetch database.

Note

Downstream Capture with ADG does not support per-PDB Extract.

Click **Next**.

Managed Options Screen

On the **Managed Options** screen, configure the auto start and auto restart options for the Extract process. The following table provides these options:

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.
 Note This option only appears while creating the Extract or Replicat and not when you set up the managed processes in the Profiles page.	
Auto Start	Enables autostart for the process.
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.

Click **Next**.

Extract Parameter File Screen

On the **Parameter File** screen, you can edit the parameter file in the text area to list the table details that you are interested in capturing. Here's a sample Extract parameter file:

```
EXTRACT exte
USERIDALIAS ggeast DOMAIN OracleGoldenGate
EXTTRAIL east/ea
DDL INCLUDE MAPPED
TABLE hr.*;
```

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 options.

You return to either the Administration Service home page or the Extract page where all created Extracts are listed.

Create a Parameter File for Extract

Follow these instructions to create a parameter file for an Extract.

1. (MySQL only) When running Oracle GoldenGate for MySQL on a Windows server and on a Linux server (for OGG version lower than 23.10.x) which is on a remote host from the database server, ensure that their time zones are the same and if not, use the SETENV(TZ) parameter within the Extract and set it to the time zone of the database server.

For TZ format and more details, refer to the SETENV parameter in the Parameters and Functions Reference Guide.

See the following links for reference:

<https://learn.microsoft.com/en-us/cpp/c-runtime-library/reference/tzset?view=msvc-170#remarks>

https://www.gnu.org/software/libc/manual/html_node/TZ-Variable.html

2. On the source system, issue the following command:
EDIT PARAMS *extract_name*

Where:

extract_name is either the name of the Extract that you created with the ADD EXTRACT command or the fully qualified name of the parameter file if you defined an alternate location when you created the group.

3. Enter the parameters in the order shown in the following table, starting a new line for each parameter statement. Some parameters apply only for certain configurations.

Parameter	Description
EXTRACT group	Configures Extract as an online process with checkpoints.
• group is the name of the Extract group that you created with the ADD EXTRACT command.	
[, USERIDALIAS alias options]	See USERIDALIAS, to specify database credentials.
ENCRYPTTRAIL algorithm	Encrypts all trails that are specified after this entry.
SOURCECATALOG	Specifies a default container in an Oracle multitenant container database or SEQUENCE statements. Enables the use of two-part names (schema.object) where three-part names otherwise would be required for those databases. You can use multiple instances of this parameter to specify different default containers or catalogs for different sets of TABLE or SEQUENCE parameters.

Parameter	Description
TABLE [container. catalog.]owner.object schema.object library/file library/file(member);	Specifies the fully qualified name of an object or a fully qualified wildcarded specification for multiple objects. If the database is an Oracle multitenant container database, the object name must include the name of the container or catalog unless SOURCECATALOG is used. • schema is the schema name or a wildcarded set of schemas. • object is the table name, or a wildcarded set of tables. • library is the IBM i library name or a wildcarded set of libraries. • file is the IBM i physical file name or a wildcarded set of physical files. • member is the IBM i physical file member name or a wildcarded set of member names. When using the IBM i native name format (library/file with optional member) the only valid wildcards are a name with at least one valid character followed by a trailing asterisk (*) or *ALL which matches any name. Note The member name is optional, and must be provided if the member names are required to be written in the trail as part of the object name. Without member names all members in a physical file be implicitly merged as a single object in the trail. See Specifying Object Names in Oracle GoldenGate Input guidelines for specifying object names in parameter files.
SCHEMAEXCLUDE	Parameters that can be used in conjunction with one another to exclude specific objects from a wildcard specification in the associated TABLE statement.
TABLEEXCLUDE	
EXCLUDEWILDCARDOBJECTSONLY	

4. Enter any appropriate optional Extract parameters listed in the Oracle GoldenGate Parameters.
5. Save and close the parameter file.

The following sample Extract parameter file explains various configuration parameters and options for Extract:

```
ADD EXTRACT extract_name
{, datasource}
{, BEGIN start_point} | {position_point}
[, PARAMS pathname]
[, REPORT pathname]
[, DESC 'description']
```

- `extract_name` is the name of the Extract group. A group name is required.
- `datasource` is required to specify the source of the data to be extracted. Use one of the following:
 - `TRANLOG` specifies the transaction log as the data source. When using this option for Oracle Enterprise Edition, you must issue the `DBLOGIN` command as the Extract database user (or a user with the same privileges) before using `ADD EXTRACT` (and also before issuing `DELETE EXTRACT` to remove an Extract group).

Use the `bsds` option for Db2 z/OS to specify the Bootstrap Data Set file name of the transaction log.
 - `INTEGRATED TRANLOG` specifies that this Extract will operate in integrated capture mode to receive logical change records (LCR) from an Oracle Database logging server. This parameter applies only to Oracle databases.
 - `EXTTRAILSOURCE trail_name` to specify the relative or fully qualified name of a local trail.
- `BEGIN start_point` defines an online Extract group by establishing an initial checkpoint and start point for processing. Transactions started before this point are discarded. Use one of the following:

- **NOW** to begin extracting changes that are timestamped at the point when the `ADD EXTRACT` command is executed to create the group or, for Extract in integrated mode, from the time the group is registered with the `REGISTER EXTRACT` command. Extract needs to be registered for Oracle and PostgreSQL databases only.

Timestamp: The format for specifying an exact timestamp as the begin point. Use a begin point that is later than the time at which replication or logging was enabled.

The following example shows the repositioning of Extract using a specific timestamp:

```
OGG (http://localhost:11000 ggeast as pdb1@east.oracle.com) 95> dblogin
useridalias ggma
```

Successfully logged into database PDB1.

```
OGG (http://localhost:11000 ggeast as ggma@ggeast/PDB1) 96> alter
extract exte , begin 2024-05-03T03:48:00Z
```

```
2024-05-03T03:50:49Z INFO    OGG-08100 Extract exte I/O position is
altered and reposition to older date and time position 2024-05-03
03:48:00.000000 current date and time position 2024-05-03
03:49:04.000000.
```

```
Duplicate transactions are filtered out. Perform output trail
ETROLLOVER if duplicate transaction output is desired, or Extract
configuration was updated.
```

```
2024-05-03T03:50:49Z INFO    OGG-08100 Extract altered.
```

- `position_point` specifies a specific position within a specific transaction log file at which to start processing. For the specific syntax to use for your database.
- `PARAMS pathname` is required if the parameter file for this group will be stored in a location other than the `dirprm` sub-directory of the Oracle GoldenGate directory. Specify the fully qualified name. The default location is recommended.

- `REPORT pathname` is required if the process report for this group will be stored in a location other than the `dirrpt` sub-directory of the Oracle GoldenGate directory. Specify the fully qualified name. The default location is recommended.
- `DESC 'description'` specifies a description of the group.

Additional Parameter Options for Extract

Learn about additional parameters that may be required for your Extract configuration.

Extract uses a database logmining server in the mining database to mine the redo stream of the source database. You can set parameters that are specific to the logmining server by using the `TRANLOGOPTIONS` parameter with the `INTEGRATEDPARAMS` option in the Extract parameter file.

Note

For detailed information and usage guidance for these parameters, see the "DBMS_CAPTURE_ADM" section in *Oracle Database PL/SQL Packages and Types Reference*.

The following parameters can be set with `INTEGRATEDPARAMS`:

- `CAPTURE_IDKEY_OBJECTS`: Controls the capture of objects that can be supported by `FETCH`. The default for Oracle GoldenGate is `Y` (capture ID key logical change records).
- `DOWNSTREAM_REAL_TIME_MINE`: Controls whether the logmining server operates as a real-time downstream capture process or as an archived-log downstream capture process. The default is `N` (archived-log mode). Specify this parameter to use real-time capture in a downstream logmining server configuration. For more information on establishing a downstream mining configuration, see [Downstream Extract for Oracle GoldenGate Deployment](#).
- `INLINE_LOB_OPTIMIZATION`: Controls whether LOBs that can be processed inline (such as small LOBs) are included in the LCR directly, rather than sending LOB chunk LCRs. The default for Oracle GoldenGate is `Y` (Yes).
- `MAX_SGA_SIZE`: Controls the amount of shared memory used by the logmining server. The shared memory is obtained from the streams pool of the SGA. The default is 1 GB.
- `PARALLELISM`: Controls the number of processes used by the logmining server. The default is 2. For Oracle Standard Edition, this must be set to 1.
- `TRACE_LEVEL`: Controls the level of tracing for the Extract logmining server. For use only with guidance from Oracle Support. The default for Oracle GoldenGate is 0 (no tracing).
- `WRITE_ALERT_LOG`: Controls whether the Extract logmining server writes messages to the Oracle alert log. The default for Oracle GoldenGate is `Y` (Yes).

See [Managing Server Resources](#).

Register an Extract from the Admin Client

Registering an Extract is required for change synchronization for Oracle and PostgreSQL databases.

This topic describes registering an Extract from the Admin Client. You can also register an Extract from the web interface while creating the Extract. See [Add an Online Extract](#) to see the steps to register an Extract from the web interface.

In case you need to upgrade Oracle GoldenGate and the installation directory for the Extract process is different, you need to use the MIGRATE option with the REGISTER EXTRACT command in Admin Client. See REGISTER EXTRACT in the *Command Line Interface Reference for Oracle GoldenGate* to know more.

Register Extract for Oracle

Follow these instructions to register an Extract. Extract registration must be done prior to creating an Extract.

Ensure that you are connected with the database using the DBLOGIN command.

See REGISTER EXTRACT in the *Command Line Interface Reference for Oracle GoldenGate* for more information.

1. Using the Admin Client, connect to the deployment, then connect to the credential alias for the source database.

```
OGG> CONNECT https://remotehost:svrmgrport DEPLOYMENT deployment_name AS  
deployment_user PASSWORD deployment_password
```

When running the CONNECT command, the command prompt changes from "not connected" to "https://servername:port deployment_name", as shown in the following example:

```
OGG (https://pdbeast.vcn.oracle.com:16000depl_east)>
```

2. Connect to the database using the DBLOGIN command:

```
OGG (https://remotehost:portoracle_source)> DBLOGIN USERIDALIAS alias
```

3. Register the Extract. The Extract names cannot be more than 8 alpha-numeric characters.

```
OGG (https://remotehost:portoracle_source)> REGISTER EXTRACT extname  
DATABASE
```

You can also register an Extract in the background while creating an Extract from the Oracle GoldenGate MA web interface. See [Add an Online Extract](#) for details.

Register Extract for PostgreSQL

An Extract for PostgreSQL must be registered with the database and be granted a reserved replication slot. Replication slots are allocated through the database configuration setting max_replication_slots and can be configured as discussed in [Database Configuration](#).

Follow these instructions to register an Extract. Extract registration must be done prior to creating an Extract. See REGISTER EXTRACT in the *Command Line Interface Reference for Oracle GoldenGate* for more information.

1. Connect to the deployment, then connect to the credential alias for the source database.

```
CONNECT https://remotehost:svrmgrport DEPLOYMENT
      deployment_name AS deployment_user PASSWORD deployment_password
```

```
OGG (https://remotehost:16000postgresql_source)> DBLOGIN USERIDALIAS alias
```

2. Register the Extract, which internally creates a replication slot for the Extract. Extract names cannot be more than 8 alpha-numeric characters.

```
REGISTER EXTRACT extname
```

To explicitly register Extract with the pgoutput plugin, you must specify the pgoutput plugin type:

```
REGISTER EXTRACT extname PGPLUGINTYPE pgoutput;
```

You can also register an Extract from the Oracle GoldenGate MA web interface. See [Add an Online Extract](#).

Guidelines and Limitations for PGOUTPUT Plugin Type

Guidelines when configuring PGOUTPUT plugin type

Follow these guidelines when setting the plugin type for PostgreSQL:

- The publication(s) must be created before registering the Extract process.
- Publication names must not contain special characters: comma (,), single quote ('), and double quotes (").
- Multiple publication names can be specified as a comma-separated list within `publication_names`. For example:

```
TRANLOGOPTIONS STREAMINGOPTIONS
(publication_names="my*123,my_publication2,123 pub,my_publication");)
```

For details, see `TRANLOGOPTIONS STREAMINGOPTIONS` parameter.

- The final publication name in the list must end with a semicolon (;).

Limitations

The following restrictions apply while selecting a plugin type for PostgreSQL:

- **Timezone (TZ) environment variable:** The WAL sender process always transmits commit timestamps in UTC. If the OGG session timezone differs from UTC, discrepancies may occur when positioning by timestamp in the CDC Extract process. To prevent this issue, users must set the environment variable `TZ=UTC` at the deployment layer. This limitation applies specifically to the pgoutput plugin.
- **Multiple truncates:** Multiple truncate operations are only supported in pgoutput plugin. The `test_decoding` plugin type does not allow multiple truncates.
- **Tables with unique indexes but no Primary Key:** If a table does not have primary keys but contains unique indexes, then the replica identity must use indexes or it must be set to `FULL`.

- **Publication Deletion Impact on CDC Extract:** If a publication is deleted while the CDC Extract is still running, there is a risk of replication slot corruption, causing it to stop functioning. In such cases, the impacted replication slot is not usable, and recreating the same publication will not resolve the issue. The recommended solution is to create a new publication and associate it with a new replication slot to restore CDC functionality.
- **Tables eligible for publication:** Only persistent base tables and partitioned tables can be included in a publication. The following object types cannot be part of a publication:
 - Temporary tables
 - Unlogged tables
 - Foreign tables
 - Materialized views
 - Regular views
- **Bi-Directional Replication Support with pgoutput Plugin:** When using the pgoutput plugin for bidirectional replication, the checkpoint table must be created and included in the publication list before starting the CDC Extract process. Additionally, the replication slot should be created only after the checkpoint table has been created and added to the publication.
- **Column-level publication limitation:** Specifying a subset of columns for a table in a publication is not supported for CDC replication. You must include the entire table in the publication. Attempting to add only partial columns may lead to errors or data discrepancies in the CDC stream.

Configure Extract for Oracle Autonomous Database

Following are the steps to configure an Extract to capture from an Oracle Autonomous AI Database :

1. Install Oracle GoldenGate for your Oracle Autonomous AI 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 AI Database instance will be created. See [Add a Deployment](#).
3. Obtain Oracle Autonomous AI Database Client Credentials.

To establish connection to your Oracle Autonomous AI 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 AI 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 AI 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 AI Database instance.

4. Configure the server where Oracle GoldenGate is running to connect to the Autonomous AI 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 AI 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.

The connections for the default `ggadmin` user happens only with `low`, although `high` is used in the connection string, for example `graphdb1_high`.

See *Local Naming Parameters in the tnsnames.ora File* chapter in the *Oracle AI 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 AI Database Serverless* guide or [Predefined Database Services](#) in Autonomous AI Database on Dedicated Exadata Infrastructure documentation.

- 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 AI Database).

Note

Oracle Autonomous AI Database allows connections on ports 1521 and 1522. However, the Oracle GoldenGate connection only works with the port 1522. This is mandatory. If you use 1521, you get the following error:

```
Login failed. OCI Error ORA (status = 24459-ORA-24459:
OCISessionGet() timed out waiting for pool to create new
connections\\n)", "code": "OGG-08110"
```

Sample Connection String

```
adw1_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
```

You can also use the following command with `high` in the connection string, but it will still connect with `low`:

```
ALTER CREDENTIALSTORE ADD USER
ggadmin@dbgraph1_high 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 [Enable Minimal Supplemental Logging in Oracle Autonomous AI Database for Oracle GoldenGate Extract](#).

9. Add heartbeat table.

```
ADD HEARTBEATTABLE
```

10. Add and configure an Extract to capture from the Oracle Autonomous AI Database instance. See [Add an Online Extract](#) for steps to create an Extract.

Oracle GoldenGate Extract is designed to work with the Oracle Autonomous AI 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 AI 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 AI 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 AI 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 AI Database instance. Here's an example:

```
START EXTRACT exte
```

This completes the process of configuring an Extract for Oracle Autonomous AI Database and you can use it like any other Extract process.

Downstream Extract for Oracle GoldenGate Deployment

A downstream Oracle GoldenGate deployment allows you to offload the processing of the redo logs to an alternative database referred to as a mining database. The mining database does not need to contain any data from the source database, but must be the same database version or higher. This means Oracle version 12c as a source database can use mining

database version 19c or higher allowing Oracle GoldenGate to capture changes from an Oracle 12c database without actually running on the Oracle 12 database.

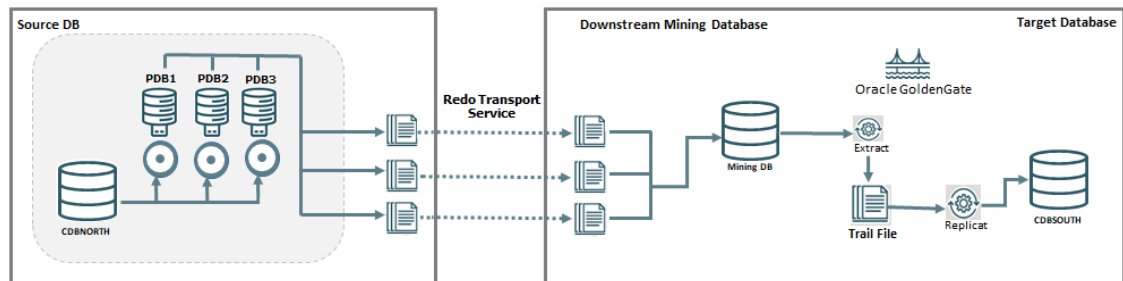
Note

The source and downstream database servers must have the same endianness and processor types. For example, downstream Extract will work with source Linux (x86) to a downstream Windows (x86) server, but it does not work if the source server is HP-UX but the downstream server is AIX, or the source server is HP PA-RISC but the downstream server is HP-Itanium.

A downstream mining database accepts archived logs and online redo logs from a source database. This implies that a downstream Extract can exist in two modes:

- **Real-Time Mining mode:** In Real-time mining mode, the redo records are processed from the online or standby redo logs immediately, so, there is no delay in processing the database changes.
- **Archive Mining mode:** In archive mining mode, the redo records are not processed until they are archived into a redo log file.

In a typical downstream deployment, the source database, where changes are made to the data, will ship its redo logs or archive logs to a mining database where the Extract process will mine the redo logs and write out the trail files. Downstream Extract can also be used in situations where Oracle GoldenGate cannot establish a connection with the source database. In the following diagram, **CDBNORTH** is the source database and **CDBSOUTH** is the target database. There is a **mining database** between the source and target with the database logging server that transfers redo logs from the downstream Extract to the target.

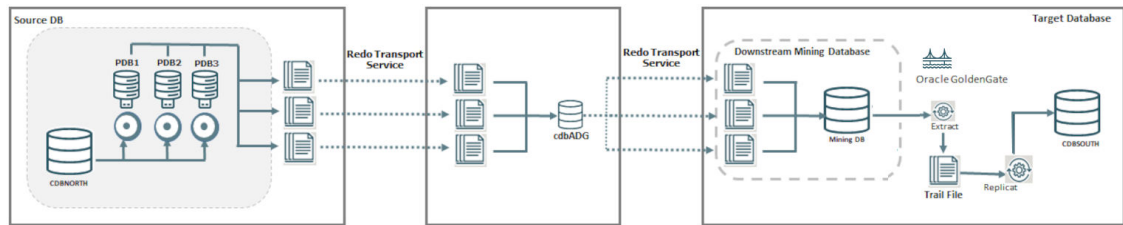


An alternative option is to use a cascading approach. If there is an Active Data Guard standby, you can have the standby ship the redo logs to the mining database instead of shipping the logs from the primary. This option should be used if there is limited network bandwidth and the standby database is in the same region or data center as the mining database.

This approach uses an ADG configured in a cascaded mode to transport redo logs to a downstream mining database, which is then referred by a downstream Extract, reducing the overhead on the source database, as shown in the following diagram.

Note

The Extract process for Oracle cannot connect to a Data Guard standby or Active Data Guard standby to capture data.



For configuration details about setting up a cascaded downstream environment, see [Enable Downstream Extract to Work with ADG](#).

Configure the Source and Downstream Mining Databases

In this section, you will learn about the parameters and user management configurations required on database side.

Oracle GoldenGate User in Source and Downstream Mining Databases

The Oracle GoldenGate Administrator user needs to exist on each database. In the container databases, a common user is required as the Extract will be used within the Root container.

To set up users for Oracle AI Database 26ai and higher, the OGG_CAPTURE and OGG_APPLY user role needs to be assigned to the new user. Refer to [Grant User Privileges for Oracle AI Database 26ai and Higher](#) for details.

To set up users for Oracle Database 21c and lower, PL/SQL packages are used. The following example shows the creation of a user and the associated privileges granted to this user when using Oracle GoldenGate with Oracle Database 21c and lower:

```
CREATE USER c##ggadmin IDENTIFIED BY password CONTAINER = all;
ALTER USER c##ggadmin SET CONTAINER_DATA = all CONTAINER = current;
GRANT ALTER SESSION TO c##ggadmin CONTAINER = all;
GRANT connect, resource TO c##ggadmin CONTAINER = all;

BEGIN
  DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE
  ( grantee          => 'C##GGADMIN'
  , privilege_type    => 'CAPTURE'
  , grant_select_privileges => TRUE
  , container         => 'ALL'
  );
END;
/
```

Note

The common user (c##ggadmin) is required for the downstream database only.

Configure Database Connections

Database Net Service Requirements

The source and downstream databases can be accessed using the Database Net Services in both directions.

In Oracle GoldenGate, connections for the source database (**CDBNORTH_root**) and the downstream database (**cdbDSC_root**) can be set up. The source database connection is only needed to store dictionary data, during Extract registration. You can use the **FETCHUSERIDALIAS** parameter to connect to the downstream mining database (**cdbDSC_root**) for fetches at runtime. With ADG redirection, the registering of Extract can be redirected to the source from an ADG. For setting up connections, make sure that you have the database credentials for these users, to create credentials from the Oracle GoldenGate deployment.

You can test the Net Service with SQL*Plus, which is part of the Oracle GoldenGate shiphome.

Source Database Prerequisites

For configuring database connections, make sure that the following settings are done on the source database instances:

- All databases must include:

```
ALTER SYSTEM SET ENABLE_GOLDENGATE_REPLICATION = TRUE;
```
- The source databases must be in archive log mode. To enable archiving on the database, issue the following command that tells the database to create an archive log when switching redo logs.

```
ALTER DATABASE ARCHIVE LOG;
```
- (Recommended) All databases must be in Force Logging mode.

```
ALTER DATABASE FORCE LOGGING;
```
- Source database must have minimal supplemental logging enabled that includes the **TRANSDATA**/**SCHEMATRANDATA** during the offloading of the Extract process to the Mining database because this is where the transactions are occurring.
- The password files (**orapwSID**) of the source and downstream database are identical. So, the **remote_login_passwordfile** must be set to **SHARED** or **EXCLUSIVE**.

Downstream Mining Database Prerequisites

For configuring database connections, make sure that the following settings are done on the source and downstream database instances:

- All databases must include:

```
ALTER SYSTEM SET ENABLE_GOLDENGATE_REPLICATION = TRUE;
```
- **STREAMS_POOL_SIZE** must be set in Oracle 21c and lower. For example

```
ALTER SYSTEM SET STREAMS_POOL_SIZE = 1G;
```
- The mining databases must be in archive log mode. To enable archiving on the database, issue the following command that tells the database to create an archive log when switching redo logs.

```
ALTER DATABASE ARCHIVE LOG;
```
- The downstream database must have different settings for the following:

- DBID
- DBNAME
- DB_UNIQUE_NAME
- GLOBAL_NAME

In a downstream environment, the unique database name matches the database name.

- The password files (orapwSID) of the source and downstream database are identical. So, the remote_login_passwordfile must be set to SHARED or EXCLUSIVE.

Configure Parameters for Downstream Database

Use this section to configure parameters for real-time mining mode and archive mining mode, as needed.

Parameter Settings for Downstream Extract in Real-Time Mining Mode

For downstream Extract in Real-Time Mining mode, the primary prerequisite is that the downstream database must have standby redo log files configured correctly.

Downstream Real-Time Mining is enabled with an explicit setting within the Extract Parameter file. The configuration of the LAD is independent from this mode. If the TRANLOGOPTIONS INTEGRATEDPARAMS (REAL_TIME_MINE Y) parameter was not already set, you need to set it to enable downstream Extract for real-time mining mode:

```
TRANLOGOPTIONS INTEGRATEDPARAMS (DOWNSTREAM_REAL_TIME_MINE Y)
```

Note

There can be only one real-time Extract on the downstream database for a given source database. Other Extracts have to be non-realtime. This is a restriction from redo transport.

The following command run on the Downstream Mining database shows the local backup being set up using the FRA:

```
ALTER SYSTEM
  SET log_archive_config='DG_CONFIG=(CDBNORTH, cdbDSC)';
ALTER SYSTEM set FAL_SERVER = 'CDBNORTH_root';
```

Standby Redo Log Files must exist on the downstream database. If there are n online redo log files at the source, then there should be $n+1$ standby redo log files at the target of the same size.

Create the Standby Redo Log Files on the Downstream Mining database

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 [Creating a Physical Standby Database](#) for detailed instructions about adding standby redo log files to a database.

① Note

If there will be 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*(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 Downstream Mining 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
```

Parameter Settings for Downstream Extract in Archive Mining Mode

For setting up a downstream Extract in either Archive Mining mode or the Real-Time Mining mode, you must set up the redo transport. Include the following parameters when setting up the downstream Extract in Archive-Log-Only mode:

- `LOG_ARCHIVE_CONFIG`
- `LOG_ARCHIVE_DESTINATION (LAD)`
 - `LOG_ARCHIVE_DEST_1` for local backup of all databases
 - `LOG_ARCHIVE_DEST_2` service for redo transport on the source database

Note

`LOG_ARCHIVE_DEST_2` is independent from the use of Real Time Mining.

- Local database backup destination is set using the Flash Recovery Area
- `FAL_SERVER`
- All the required system parameters can be set dynamically with `scope=both`

The Archive Log Destination (LAD) parameter settings are required for the local backup destination and redo transport services on the source databases.

Set up the LAD parameters for **local backup destination**, using the following commands:

```
ALTER SYSTEM SET db_recovery_file_dest = '/u01/app/oracle/fast_recovery_area';
ALTER SYSTEM SET db_recovery_file_dest_size = 100G;
ALTER SYSTEM SET log_archive_dest_1 = 'LOCATION=USE_DB_RECOVERY_FILE_DEST';
```

Set up the LAD parameter options for **redo transport services** on the source database using the following commands:

```
ALTER SYSTEM SET log_archive_config = 'DG_CONFIG = (CDBNORTH, cdbDSC)';
ALTER SYSTEM
    SET log_archive_dest_2='SERVICE=cdbDSC_root
                           ASYNC
                           NOREGISTER
                           VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE)
                           DB_UNIQUE_NAME=cdbDSC';
```

To learn more about the LOG_ARCHIVE_DEST parameter and its attributes, see LOG_ARCHIVE_DEST_n in the *Database Reference* guide.

Configure Oracle GoldenGate Extract for Downstream Deployment

In this section, you will learn about the parameters required by Oracle GoldenGate to set up connections with source and downstream databases and configure Extract for downstream database.

Configure Oracle GoldenGate for Downstream Database

The downstream Extract parameter file must include the following parameters for setting up a downstream deployment:

- **TRANLOGOPTIONS MININGUSERALIAS**: Sets the connection to the downstream mining database for internal metadata queries.
- **FETCHUSERIDALIAS**: Connects to the source database for fetching data that cannot be natively extracted out of the transaction log.
- **TRANLOGOPTIONS INTEGRATEDPARAMS (DOWNSTREAM_REAL_TIME_MINE Y)**: If set to Y, then Extract captures changes from the standby redo logs files at the downstream database. If not set, then Extract captures from the archive log files.

Following is an example of the Extract parameter file including these parameters:

```
EXTRACT EXTDS
USERIDALIAS cgg_CDBNORTH
TRANLOGOPTIONS MININGUSERALIAS cgg_cdbDSC
TRANLOGOPTIONS INTEGRATEDPARAMS (DOWNSTREAM_REAL_TIME_MINE Y)
FETCHUSERIDALIAS cgg_CDBNORTH DOMAIN OracleGoldenGate

EXTTRAIL Downstream/ea
SOURCECATALOG CDBNORTH_PDB1
DDL INCLUDE MAPPED
TABLE HR.*;
```

Follow the given steps to set up the Oracle GoldenGate deployment for downstream Extract:

1. Add credentials for the source database:

```
ALTER CREDENTIALSTORE ADD USER c##ggadmin@CDBNORTH_root ALIAS cgg_CDBNORTH  
DOMAIN OracleGoldenGate PASSWORD password
```

2. Add credentials for the downstream database:

```
ALTER CREDENTIALSTORE ADD USER c##ggadmin@cdbDSC_root ALIAS cgg_cdbDSC  
DOMAIN OracleGoldenGate PASSWORD password
```

3. Use DBLOGIN to connect to the source database and downstream database from Oracle GoldenGate:

```
DBLOGIN USERIDALIAS cgg_CDBNORTH
```

```
MININGDBLOGIN USERIDALIAS cgg_cdbDSC
```

In this example, `cgg_CDBNORTH` is the credential alias for the source database **CDBNORTH** and `cgg_cdbDSC` is the credential alias for the mining database.

4. Add and register an Extract, edit the parameter file to include the Extract parameters provided previously, add the Extract trail name and location, start the Extract:

```
ADD EXTRACT extdsc, INTEGRATED TRANLOG, BEGIN NOW  
REGISTER EXTRACT extdsc, DATABASE CONTAINER (CDBNORTH_PDB01)
```

```
ADD EXTTRAIL Downstream/ea, EXTRACT extdsc  
START extdsc
```

Configure Cascaded Downstream Extract Using Active Data Guard

If a connection to the source database is not possible, GoldenGate can be configured to use an Active Data Guard (ADG) standby database. The standby database is available for read-only connections to perform metadata queries and fetching. In this section, you will learn how to configure a Downstream Extract using ADG and without establishing a connection with the source database.

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) for fetching and metadata queries.

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 EXTDS
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' = (CDBNORTH, cdbADG_src,
cdbDSC)';
```

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 as shown in the following example:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='location= /home/arc_dest/dstrm
VALID_FOR=(STANDBY_LOGFILE,ALL_ROLES)' scope=both
```

where /home/arc_dest/dstrm is the location for foreign archive logs. Do not use FRA for storing foreign archive logs.

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.

Extract Actions

Extract actions include tasks like monitoring details for the Extract, checkpoint details, DDL/DML statistics, cache manager statistics, and other details.

Use the **Actions** column to start or stop the Extract or view and manage its details. When you click the **Details** icon for an Extract, you can perform the following tasks for it.

When you change the status, the list options change accordingly. As status changes, 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.

Access Extract Details

From the Extract section of the Administration Service Overview page, click **Action**, **Details** for the specific Extract to view its details. The following tabs are displayed:

- **Process Information:**

The status of the selected Extract process including the type, credentials, and trail details including trail name, trail subdirectory, trail sequence, and trail size.

- **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.

- **Cache Manager Statistics:**

Access the global statistics and object pool statistics information for the Extract process from this page.

- **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.

- **Integrated Diagnostics:**

Integrated Diagnostics feature is responsible for collecting diagnostic and performance data for each of the Oracle GoldenGate replication components and sub-components. This process preserves interval data in memory and computes all statistical values related to CPU usage, session wait times, various counts and status. See [About Integrated Diagnostics](#).

Start or Stop Extract

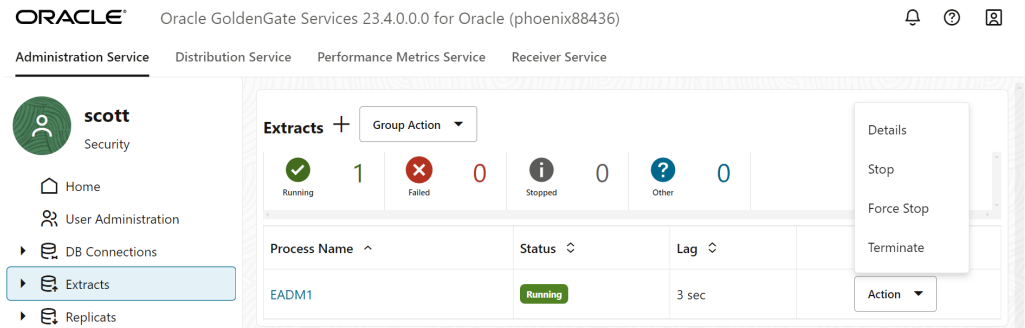
From the Administration Service web page, click Extract to open the Extracts information web page, where all Extracts are listed. From the Action column, you can select the **Start/Stop** icon depending on the current state of the Extract.

If the Extract is in abended state, it displays with a yellow icon. A green icon indicates that the Extract is running and a red icon indicates Extract is in stopped state.

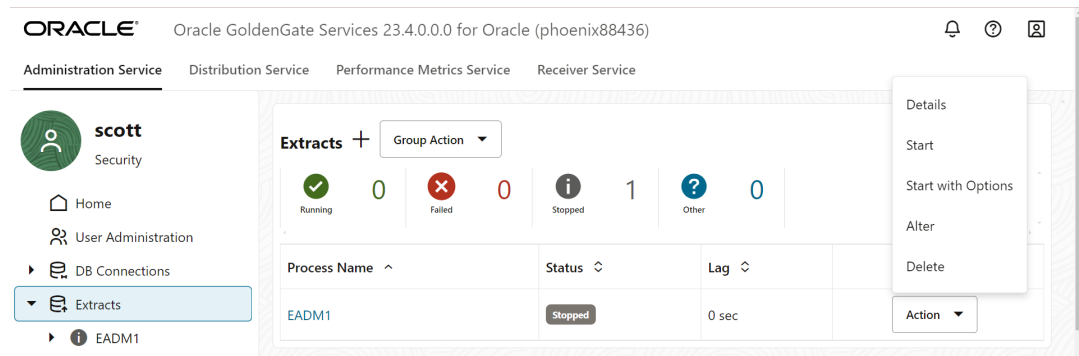
Alter Extract

To alter an Extract:

1. Stop the Extract:
 - a. From the Administration Service left navigation pane, click **Extracts**.
 - b. Click **Action** from the Extracts and then click **Stop**.



- After the Extract stops, click Action again. The Alter option is displayed.



- Click Alter. The Alter Extract dialog box is displayed. This option allows you to change when the Extract begins. It does not start the Extract.
- Select the start option for the Extract from the drop down list. The available options are:
 - Begin Now: Use this option to immediately start the Extract.
 - Custom Time: Use this option to set a time for starting the Extract.
 - CSN: Use this option to set a CSN value for starting the Extract.

Delete Extract

To delete an Extract:

- Stop the Extract using the **Actions, Stop** option from the Extract section of the Administration Service Overview page.
- Click **Delete** to remove the Extract.

Note

The **Delete** option appears only when the Extract is in **Stopped** state.

Positioning Extract to a Specific Start Point

You can position the Extract to a specific start point in the transaction logs. You can set this from the web interface, command line interface (Admin Client), or the REST API service endpoint.

If you are creating an Extract then you must perform the following steps to start the newly created Extract at a specific position in the database:

1. Add the Extract with the `BEGIN NOW` option, as shown in the following example:

```
ADD EXTRACT extn, TRANLOG, BEGIN NOW
```

2. Register the Extract.

```
REGISTER EXTRACT extn DATABASE
```

3. Alter this Extract and specify the timestamp.

```
ALTER EXTRACT extn BEGIN 2020-08-02T06:05:30.000Z
```

Web Interface

To position an Extract to a specific start point in the database:

1. Stop the Extract if it's running.
2. From the Actions column, click the three-dot icon and click **Alter**.
3. From the **Alter Extract** dialog box, click the **Begin** option and select **Custom Time**.
4. Select the timestamp from the **Custom Time** field and select the **Timezone**.
5. Click **Submit**.
6. Restart the Extract by clicking the Play icon from the **Action** column.

Admin Client

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 extn, 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.

Bounded Recovery

Valid for Oracle database only.

Bounded Recovery is a component of the general Extract checkpointing facility. It guarantees an efficient recovery after Extract stops for any reason, planned or unplanned, no matter how many open (uncommitted) transactions there were at the time that Extract stopped, nor how old they were.

Bounded Recovery sets an upper boundary for the maximum amount of time that it would take for Extract to recover to the point where it stopped and then resume normal processing.

The default parent directory is BR, which is in the root directory that contains the Oracle GoldenGate installation files.

The default location of the parent BR directory in Oracle GoldenGate Microservices Architecture is: \$OGG_HOME/var/run/BR. BR requires at least twice BRINTERVAL log retention to guarantee a Bounded Recovery. BRINTERVAL default is 4 hours.

Caution

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

For parameter and syntax details, see BR.

Modifying the BR Parameter

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

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

What to Do if Extract Abends During Bounded Recovery

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

If the problem is not correctable, attempt to restart Extract with the BRRESET option. BRRESET requires Extract to be restarted manually. BRRESET is not a valid parameter in a .prm file.

Extract will recover in normal recovery mode and then turn on Bounded Recovery again.

Circumstances that Change Bounded Recovery to Normal Recovery

Most of the time, Extract uses normal recovery and not Bounded Recovery, the exception being the rare circumstance when there is a long running transaction. Certain abnormal circumstances may prevent Extract from switching from Bounded Recovery back to normal recovery mode. These include, but are not limited to, such occurrences as physical corruption of the disk (where persisted data is stored for long-running transactions), inadvertent deletion of the Bounded Recovery checkpoint files, and other actions or events that corrupt the continuity of the environment. There may also be more correctable reasons for failure.

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

Bounded Recovery is not invoked under the following circumstances:

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

After completion of normal recovery BR is turned on for runtime.

Note

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

Instantiate

Learn about different methods of performing precise instantiation for different databases.

About Instantiation

Instantiation is the process of copying source objects to the target, to be able to start replication seamlessly with uninterrupted source database operations, complete data integrity, and error-free processing at the target database.

If an Extract process captures changes from the source database, you must instantiate the corresponding database objects at the target database before replication can begin. The target database must contain a copy of the source database objects to apply changes correctly, and both source and target objects must be consistent at a specific point in time. This point of consistency may be in the past. As long as the Extract was created beforehand, replication can start by capturing changes from older transaction logs (archive log files) and catching up with the backlog. The Replicat process will then begin applying changes from the exact moment when the source and target objects were consistent. However, changes earlier than that Instantiation time (Instantiation SCN) will be ignored, changes from that time and later will be applied.

In many cases, replication involves creating an exact copy of the source objects on the target side. However, there are scenarios where replication includes data modifications or transformations. In such cases, the target objects must be adjusted to accommodate these replication requirements.

Precise Instantiation for Db2 LUW

During the initialization of the Oracle GoldenGate environment for Db2 LUW, you need to perform initial data synchronization and start Oracle GoldenGate processes for the first time. In conjunction with these procedures, you need to create process groups. To create an Extract group, an initial start position must be established in the transaction log. This initial read position is on a transaction boundary that is based on one of the following:

- End of the transaction file
- A specific LRI value

The start point is specified with the `BEGIN` option of the `ADD EXTRACT` command.

When the Extract process starts for the first time, it captures all the transaction data that it encounters after the specified start point, but none of the data that occurred *before* that point. This can cause partial transactions to be captured if open transactions span the start point.

To ensure initial transactional consistency:

Initialize the Extract process at a point in time when the database is in a paused state to avoid the capture of partial transactions. Db2 LUW provides a `QUIESCE` command for such a purpose. This is the only way to ensure transactional consistency.

Note

After the Extract is past the initialization, subsequent restarts of the Extract do not extract partial transactions, because the process uses recovery checkpoints to mark its last read position.

To view open transactions:

IBM provides a utility called `db2pd` for monitoring Db2 databases and instances. You can use it to view information about open transactions and to determine if any of them span the start point. However, because Db2 LUW log records lack timestamps, it might not be possible to make an accurate assessment. If possible, quiesce the database prior to initialization of Oracle GoldenGate.

Precise Instantiation for Oracle GoldenGate for Db2 for i

Learn about configuring an initial load of target data, adding the required processes to instantiate replication, and perform the instantiation. The expected outcome of these steps is that source-target data is made consistent (known as the initial synchronization), and that Oracle GoldenGate captures and delivers ongoing transactional changes so that consistency is maintained going forward.

During the initialization of the Oracle GoldenGate environment, you will be doing an initial data synchronization and starting the Oracle GoldenGate processes for the first time. In conjunction with those procedures, you will be creating the Extract and Replicat process groups.

To create an Extract process group, an initial start position for data capture must be established. This initial position will be based on a transaction boundary that is based on either of the following:

- a timestamp
- the end of the journal(s)
- A specific system sequence number
- A specific sequence number in the journal(s)

When Extract starts for the first time to begin capturing data, it captures all of the transaction data that it encounters after the specified start point, but none of the data that occurred before that point. To ensure that Extract does not start in the middle of ongoing transactions that would be discarded, set the tables that are to be captured to an inactive state. You can either put the system into a restricted state by using the `ALCOBJ` command to lock the objects or libraries, or you can force all of the current transactions on those tables to stop at a certain point.

After initialization is complete, remember to unlock any objects that you locked. To do so, log off of the session that locked the objects or use the `DLCOBJ` command from the IBM i command line.

Prerequisites for Instantiating for Db2 for i

These steps must be taken before starting any Oracle GoldenGate processes or native database load processes.

Configure Change Capture and Delivery

By the time you are ready to instantiate the replication environment, all of your Extract and Replicat process groups must be configured with completed parameter files as directed in [Add an Online Extract](#) and Add a Replicat topics.

Add Collision Handling

If the source database will remain active during the initial load, collision-handling logic must be added to the Replicat parameter file. This logic handles conflicts that occur because static data is being loaded to the target tables while Oracle GoldenGate replicates transactional changes to those tables.

To handle collisions, add the `HANDLECOLLISIONS` parameter to the Replicat parameter file to resolve:

- INSERT operations for which the row already exists.
- UPDATE and DELETE operations for which the row does not exist.

For more information about this parameter, see the Oracle GoldenGate *Windows and UNIX Reference Guide*.

Prepare the Target Tables

The following are suggestions that can make the load go faster and help you to avoid errors.

- Data: Make certain that the target tables are empty. Otherwise, there may be duplicate-row errors or conflicts between existing rows and rows that are being loaded.

- Constraints: If you have not done so already, disable foreign-key constraints and check constraints. Foreign-key constraints can cause errors, and check constraints can slow down the loading process.
- Indexes: Remove indexes from the target tables. Indexes are not necessary for the inserts performed by the initial load process and will slow it down significantly. You can add back the indexes after the load is finished.
- Keys: To use the `HANDLECOLLISIONS` function to reconcile incremental data changes with the load, each target table must have a primary or unique key. If you cannot create a key through your application, use the `KEYCOLS` option of the `TABLE` and `MAP` parameters to specify columns as a substitute key for Oracle GoldenGate's purposes. If you cannot create keys, the affected source table must be quiesced for the load.

Precise Instantiation Using Db2 for i Database Utilities

The following topics describe the steps to perform instantiation using the Db2 for i database utilities.

Precise Instantiation Using Db2 for i Database Utilities

For an initial load between two Db2 for i source and target systems, you can use the Db2 for i system utilities to establish the target data. To do this, you save the file(s) that you want to load to the target by using the `SAV0BJ` or `SAVLIB` commands, and then restore them on the target using the `RST0BJ` or `RSTLIB` commands.

Another alternative is to use the Db2 for i commands `CPYT0IMPF` (Copy to Import File) and `CPYFRMIMPF` (Copy from Import File) to create files that can be used with the bulk load utilities of other databases. See the Db2 for i Information Center documentation for more details on [Copying between different systems](#).

In both cases, no special configuration of any Oracle GoldenGate initial-load processes is needed. You use the change-synchronization Extract groups configured for the Db2 for i and start the change-synchronization Extract group to capture ongoing data changes while copying and loading it to a file. When the copy is finished, you start the change-synchronization Replicat group to re-synchronize rows that were changed while the copy was being applied. From that point forward, both Extract and Replicat continue running to maintain data synchronization.

To Perform Instantiation with a Database Utility

1. Make certain that you have addressed the requirements in [Prerequisites for Instantiating for Db2 for i](#).
2. Connect to the deployment on source system and start the primary Extract or Extract groups to start change extraction.

`START EXTRACT primary Extract group name`
3. Start making the copy on the source system.
4. Wait until the copy is finished and record the time of completion.
5. View the Replicat parameter file to make certain that the `HANDLECOLLISIONS` parameter is listed. If not, edit the file and add the parameter to the file.

`EDIT PARAMS Replicat group name`

Note

Do not use the `VIEW PARAMS` or `EDIT PARAMS` command to view or edit an existing parameter file that is in a character set other than that of the local operating system (such as one where the `CHARSET` option was used to specify a different character set). View the parameter file from outside Admin Client if this is the case; otherwise, the contents may become corrupted.

6. On the target system, start change replication.

```
START REPLICAT Replicat group name
```

7. On the target system, issue the following command to verify the status of change replication.

```
INFO REPLICAT Replicat group name
```

8. Continue to issue the `INFO REPLICAT` command until you have verified that change replication has posted all of the change data that was generated during the initial load. Reference the time of completion that you recorded. For example, if the copy stopped at 12:05, make sure change replication has posted data up to that point.
9. On the target system, issue the following command to turn off the `HANDLECOLLISIONS` parameter and disable the initial-load error handling.

```
SEND REPLICAT Replicat group name, NOHANDLECOLLISIONS
```

10. On the target system, edit the Replicat parameter file to remove the `HANDLECOLLISIONS` parameter. This prevents `HANDLECOLLISIONS` from being enabled again the next time Replicat starts.

Caution

Do not use the `VIEW PARAMS` or `EDIT PARAMS` command to view or edit an existing parameter file that is in a character set other than that of the local operating system (such as one where the `CHARSET` option was used to specify a different character set). View the parameter file from outside Admin Client if this is the case; otherwise, the contents may become corrupted.

11. Save and close the parameter file.

From this point forward, Oracle GoldenGate continues to synchronize data changes.

Start Extract During Instantiation

When Extract starts for the first time to begin capturing data during the instantiation process, it captures all of the transaction data that it encounters after the specified start point, but none of the data that occurred before that point. To ensure that Extract does not start in the middle of ongoing transactions that would be discarded, set the tables that are to be captured to an inactive state. You can either put the system into a restricted state by using the `ALCOBJ` command to lock the objects or libraries, or you can force all of the current transactions on those tables to stop at a certain point.

After initialization is complete, remember to unlock any objects that you locked. To do so, log off of the session that locked the objects or use the `DLCOBJ` command from the `OS/400` command line.

Positioning Extract After Startup

You may at some point, over the life of an Extract run, need to set the position of Extract in the data stream manually. To reposition Extract, use the `ALTER EXTRACT` command in GGSCI. To help you identify any given Extract read position, the `INFO EXTRACT` command shows the positions for each journal in an Extract configuration, including the journal receiver information. For more information about these commands, see Reference for Oracle GoldenGate for Windows and UNIX.

Note

Because the extract will be synchronizing all of the journals in the extract by system sequence number because it is possible for a transaction to be split across them, if a given journal is independently repositioned far into the past, the resulting latency from reprocessing the entries will cause the already-read journals to stall until the reading of the latent journal catches up.

Making the Instantiation Procedure More Efficient

The following are some suggestions to make the instantiation process more efficient.

Share Parameters Between Process Groups

Some of the parameters that you use in a change-synchronization parameter file also are required in an initial-load Extract and initial-load Replicat parameter file. To take advantage of the commonalities, you can use any of the following methods:

- Copy common parameters from one parameter file to another.
- Store the common parameters in a central file and use the `OBEY` parameter in each parameter file to retrieve them.
- Create an Oracle GoldenGate macro for the common parameters and then call the macro from each parameter file with the `MACRO` parameter.

Use Parallel Processes

You can configure parallel initial-load processes to perform the initial load more quickly. It is important to keep tables with foreign-key relationships within the same set of processes. You can isolate large tables from smaller ones by using different sets of processes, or simply apportion the load across any number of process sets.

Precise Instantiation for Oracle

There are multiple instantiation methods available for instantiating an Oracle target database. You can distinguish between these instantiation methods as *logical* or *physical*. The following table describes the properties of *logical* and *physical* instantiation methods:

Logical Instantiation	Physical Instantiation
Small Tables	Large Database
Flexible, useful for data manipulation and transformation	Fast

Logical Instantiation	Physical Instantiation
• Oracle GoldenGate Initial Load • Oracle Data Pump (expdp/impdp)	RMAN DataGuard-Split Split-Mirror Copy

Oracle GoldenGate initial load and Oracle Data Pump are *logical* instantiation methods, ideal for small tables and scenarios requiring data manipulation or transformation during the load process. In contrast, Oracle RMAN provides a physical, block-based instantiation method, offering faster performance for large-scale data transfers.

Using Oracle Data Pump for Automatic Per Table Instantiation

Oracle GoldenGate uses the Oracle Data Pump utility to create new instances of a schema, table, user for the Oracle database. It works by unloading data and metadata into a set of operating system files that are called a *dump file set*.

This method uses the Oracle Data Pump utility to establish the target data. To run this method, you start Extract, the data pumps, and replicates at the SCN at which the copy stopped. Transactions that were included in the copy are skipped to avoid collisions from integrity violations. From the process start point, Oracle GoldenGate maintains data synchronization.

No initial-load Oracle GoldenGate processes are required for this method.

You can automatically instantiate per table CSN filtering for Oracle Database with Oracle data pump, which avoids having all of your tables at the same SCN.

On the Source Database

1. Use ADD TRANDATA and ADD SCHEMATRANDATA. ADD TRANDATA/SCHEMATRANDATA.PREPARECSN automatically prepares the tables at the source so the Oracle data pump export dump file includes instantiation CSNs. Replicat uses the per table instantiation CSN set by the Oracle data pump (on import) to filter out trail records.

Use INFO TRANDATA to make sure that your table is prepared for instantiation and at what point it was done. Here's a sample of the report file:

```
2025-09-29 15:30:00 INFO OGG-10154 Schema level PREPARECSN set to mode
NOWAIT on schema
      hr
```

2. Stop Replicat on the target database.
3. Start Extract with the correct TABLE statement.

The EXPORT datapump option FLASHBACK_SCN is not needed as the tables have been prepared earlier.

On the Target Database

1. Import your exported tables using Oracle data pump, which populates system tables and views with instantiation SCNs, as well as the specified table data.
2. Start Replicat using one of the following:

Set the DBOPTIONS ENABLE_INSTANTIATION_FILTERING parameter in the Replicat parameter file to enable table-level instantiation filtering.

You can remove this parameter when replicat has processed all transactions beyond the instantiation SCN.

For all other Replicats, set the `DBOPTIONS source_dbase_name global_name` parameter in the Replicat parameter file where `global_name` is the global name of the Oracle source database that the trail is coming from.

Note

When the source has no DOMAIN, do not specify a DOMAIN for the downstream database.

Replicat queries the instantiation SCN on any new mapping and filter records accordingly. For example, see the following report file output:

```
2025-06-29 17:12:39 INFO OGG-10155 Oracle GoldenGate Delivery for Oracle,
r1.prm:
Instantiation CSN filtering is enabled on table HR.EMPLOYEES at CSN
1,851,797.
```

You can use other methods for instantiation instead of using the data pump to export and import tables also. One such method is using the `create table as a select` command or RMAN. Its steps are:

1. Use `create table` with an `at SCN` of parameter, using the following command:

```
SET_INSTANTIATION_CSN SCN for object from global_name
```

For example:

```
SET_INSTANTIATION_CSN 1 FOR u1.t1 FROM DBS1.REGRESS.RDBMS.DEV.US.Oracle.COM
```

2. If you want to remove the manual setting of the instantiation CSN later, you can use the following command:

```
CLEAR_INSTANTIATION_CSN for object from global_name
```

Using Recovery Manager (RMAN) for Instantiation

The Recovery Manager (RMAN) method of instantiating a target Oracle database is suitable for providing a physical, block-based instantiation which offers faster performance for large-scale data transfers. The RMAN process creates the target database by restoring a backup of the source database.

For large databases that are in the terabyte range, logical instantiation methods can be time-consuming and inefficient. In such cases, Oracle RMAN is a preferred solution for creating a physical clone of the source database.

The following steps outline the tasks performed during a RMAN-based instantiation for target Oracle database:

1. Use a backup of the source database.
2. Restore the backup to a new auxiliary (cloned) database.
3. Perform media recovery on the cloned database to bring it to a consistent state.

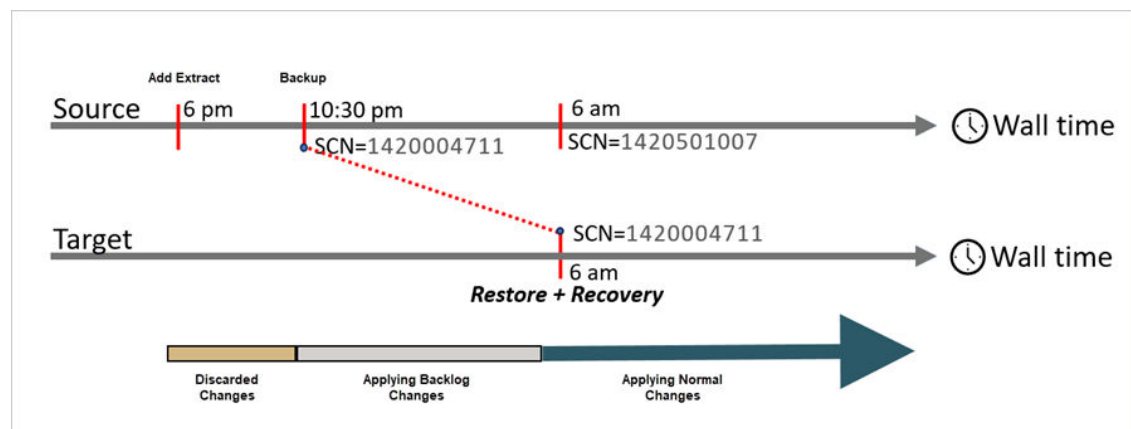
4. Configure and adjust the cloned database to meet the specific requirements of the target environment.

When using RMAN cloning for Oracle GoldenGate, it's important to manage the timing of replication to avoid data loss and ensure seamless integration. The key is to identify the point in time (or SCN) when the target database was consistent, so that both the source and target databases stay in sync.

To do this, start the Oracle GoldenGate Extract process on the source database before the backup is taken. If you find a transaction that has an older timestamp than when you registered Extract, you must wait until that transaction has completed. This is important as Extract only processes complete transactions. After the target system is set up through cloning, direct the replication changes to the target system using the DISTPATH and add the Replicat process. When you start the Replicat, make sure to start exactly at the SCN where the target database was recovered.

The following example explains the use of SCN when cloning a database using the backup from the source database.

On the source database, the backup, which is scheduled at 10 PM, is running. The backup includes database files and archive logs, which are backed up more frequently. The backup finishes at 10:30 PM with an SCN of 1420004711. This backup is then used to clone (restore and recover) the new target database. File copying, restore, and recovery take several hours, after which, the new target database is ready. At this point, both the source and target databases are consistent, as shown in the following diagram.



As time passes, new changes continue to occur on the source database, and its SCN increases. To keep the target database in sync, Oracle GoldenGate's Extract process captures changes starting from a point before the backup was taken.

Even though the Replicat process is not started immediately, when it is, it will ignore any changes that occurred before the STARTCSN (the SCN to which the target database was recovered). After that point, Replicat begins applying the backlog of changes. After Replicat is in sync, replication transitions into the normal, real-time phase.

Refer to the Duplicating Databases topic in the *Oracle Database Backup and Recovery User's Guide*, which provides detailed steps to set up a cloned target database using RMAN.

Precise Instantiation for Oracle Using the RMAN Method

The following steps are required to set up the RMAN instantiation for Oracle database while using Oracle GoldenGate.

1. Install and set up Oracle GoldenGate by deploying the software on both the source and target systems. Before configuring anything, make sure all database prerequisites are met. This includes enabling minimal supplemental logging on the source database, and using `ADD TRANDATA` or `ADD SCHEMATRANDATA` to ensure key columns in the replicated tables are logged.
2. Add and configure the Extract and DISTPATH processes on the source database.
3. Add and configure the Replicat process on the target database.
4. Start the Extract process and make sure that Extract starts running before any long-running transactions begin because Extract only captures complete transactions.
5. Start the DISTPATH process.
6. Do not start the Replicat process. wait until the target database has been fully recovered.
7. Use RMAN to take a backup of the source database. This backup will be used later to create the target database. The backup files need to be securely transferred to the target system.

```
RMAN target /  
RMAN> BACKUP DATABASE PLUS ARCHIVELOG
```

8. Use RMAN to create a duplicate of the source database and rename it as the target database name. For example, if the target database name is DBSOUTH, then rename the duplicate of the source database as DBSOUTH. Make sure to note the SCN (System Change Number) value during this process, as it marks the exact point in time where both databases are in sync. The following example displays the creation of a duplicate of the DBNORTH database, which is named as DBSOUTH (same as the target database).

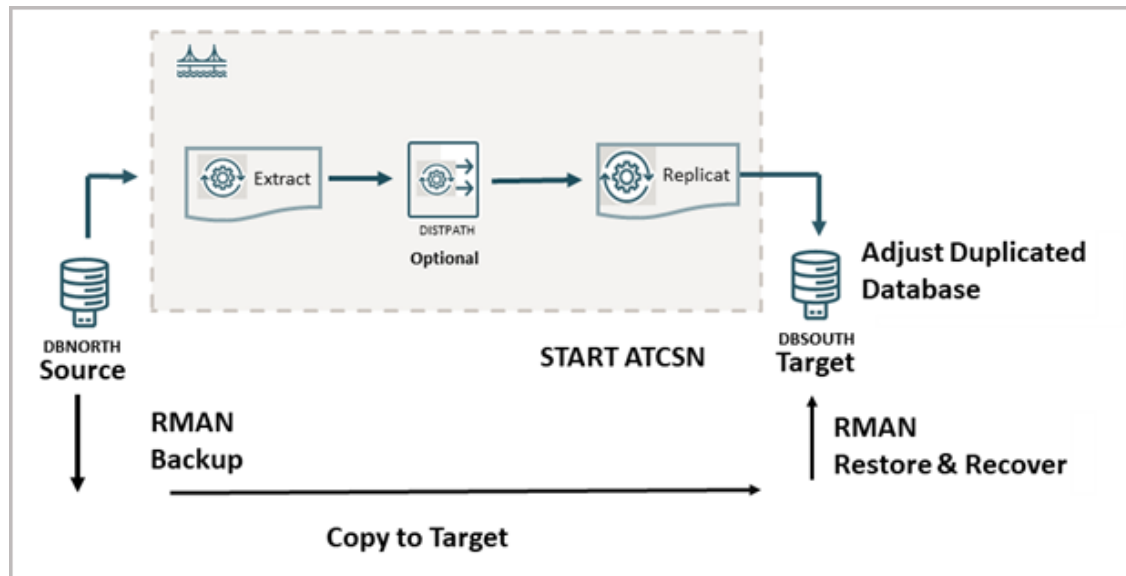
```
OS> RMAN target /  
  
RMAN> restore database preview summary  
>>> Recovery must be done beyond SCN 1420004711 to clear datafile fuzziness  
  
OS> target sys/tiger99_RR@dbnorth  
  
RMAN> connect auxiliary  
RMAN> duplicate target database to DBSOUTH nofilenamecheck until SCN  
1420004711;
```

9. Start the Replicat process on the DBSOUTH database, making sure it starts exactly at the same **SCN** as the database that was recovered using RMAN.

Use the ATSCN option to ensure there is no data gap during replication:

```
OGG http://www.example.com:12000 depl_south> START REPLICAT REPS, ATCSN  
1420004711
```

The following diagram illustrates these steps:



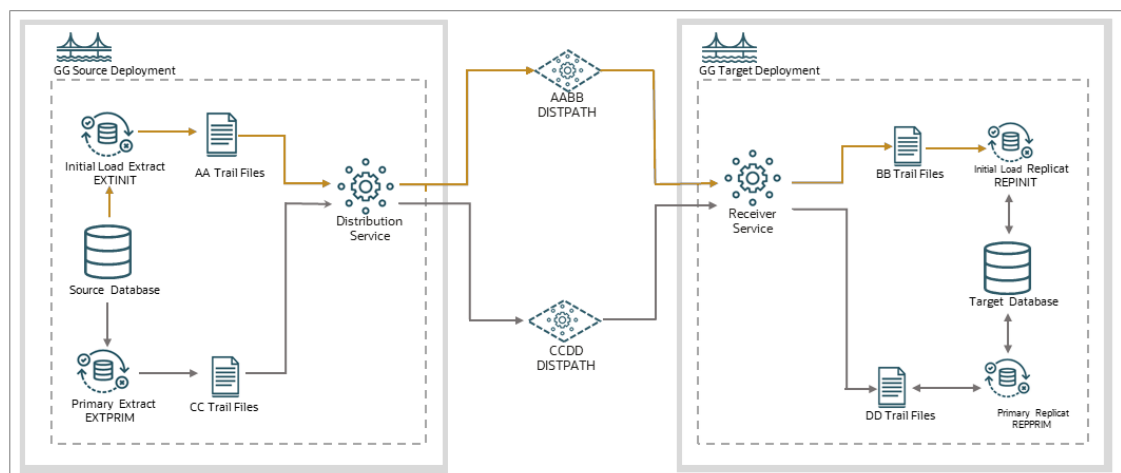
Precise Instantiation for Oracle Using Initial Load Extract

Learn about adding the Initial Load Extract using the Admin Client.

The initial load method is highly suitable for instantiation in the following scenarios:

- Loading target database with a small set of tables from the subset of the source database.
- Loading data in a heterogenous environment

The initial load Extract works by reading data directly from the source tables and writing it into an EXTFILE. It directly connects to the database and fetches the changes. As shown in the following diagram, there are two phases when performing an initial load instantiation. The EXTINIT Extract is the initial load Extract and the EXTPRM is the primary Extract. The EXTPRM process is set up but not started for continuous replication. After the initial load Extract completes instantiation, then the continuous replication process is started from the point when the initial load was initiated.



To configure this Extract, you'll require a combination of file-based initial load and change data capture (CDC) processes. The instantiation process using the initial load Extract 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

Microservices Architecture 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 (USERIDALIAS) is created.

You can use the Oracle GoldenGate web interface, Admin Client, or cURL commands to set up this configuration.

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.

Starting with Oracle GoldenGate 23ai, for initial load Extract table views will not be resolved, when resolving tables with a TABLE parameter that includes wildcards. If views need to be included in the initial load Extract, then they need to be specified in a separate TABLE parameter without any wildcards.

For example, if there is a table `hr.employees` and a view `hr.employees_view`. For initial load Extract, if you specify the following in the initial load Extract's parameter file:

```
TABLE hr.*;
```

Only the data from table `hr.employees` will be included in the initial load Extract.

If you need to include the `hr.employees_view` in the initial load, add the following parameter to the initial load Extract parameter file:

```
TABLE SCOTT.TAB1_VIEW;
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 15> ADD EXTRACT extinit  
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.oggdvops.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.oggdevops.us:9100 oggdep) 16> ADD DISTPATH ccdd SOURCE  
TRAIL://oggdep.example.com:9102/services/v2/sources?trail=CC target wss://  
dallas.oggdevops.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
```

Precise Instantiation for MySQL to MySQL Using MySQL Shell Utilities and Oracle GoldenGate

The precise instantiation method allows for the initial load of data from a source database to a target database while the source database remains online for application updates. This method ensures precise positioning of the change data capture and delivery processes without having duplicate data in the target database, and without the need to use `HANDLECOLLISIONS` in the Replicat.

Precise instantiation for Oracle GoldenGate for MySQL uses the MySQL Shell dump and dump loading utilities and is only supported between a MySQL source and a MySQL target database.

This method requires that the MySQL Shell be installed. Installation steps are available at the following link:

<https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-install.html>

For detailed information and limitations of the MySQL Shell dump and dump loading utilities, refer to the following links:

<https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-dump-instance-schema.html>

<https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-load-dump.html>

You will use the MySQL Shell dump utility to export either the MySQL instance, schema, or tables, depending on your requirements. You will then use the MySQL dump loading utility to load the objects into the target database, and afterwards configure a GoldenGate Extract and Replicat to replicate continual change data from the source to the target database.

The MySQL Shell dump utility creates multiple export files, one of which is a `@.json` file. This file is needed to determine the positioning of Extract when adding it to the source database.

The following is a sample precise instantiation method using a schema dump from the source instance, then loaded to the target instance, followed by configuration of Oracle GoldenGate for change data replication:

1. Connect to the MySQL Shell. For details about supported connection options with MySQL Shell, refer to this link: <https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-connections.html>

```
mysqlsh -mysqlx -u username -h hostname -P port
```

2. After connecting to the MySQL Shell, ensure that you are in JavaScript mode. You should see a JS > prompt. If in SQL mode, switch to JavaScript mode using the \js switch:

```
MySQL sourcehost:33060+ ssl SQL > \js
```

3. Verify that the Global Variable local_infile is set to ON.

```
MySQL sourcehost:33060+ ssl JS > \sql SHOW GLOBAL VARIABLES LIKE  
      'local_infile';
```

If the local_infile variable is set to OFF, turn it on with the following command:

```
MySQL sourcehost:33060+ ssl JS > \sql SET GLOBAL local_infile = ON;
```

4. Next, use one of the following functions to export either the instance, schema, or tables that you want to export.

```
util.dumpInstance(outputUrl[, options])  
util.dumpSchemas(schemas, outputUrl[, options])  
util.dumpTables(schema, tables, outputUrl[, options])
```

In this example, we will dump the hr schema, into the ~/dumps/hr-schema-src folder:

```
MySQL sourcehost:33060+ ssl JS > util.dumpSchemas(["hr"], "~/dumps/hr-  
schema-src")
```

5. After the export completes, use the util.loadDump utility to import the schema to the target instance.

Note

If your target instance is on a different system than the source, you'll need to copy the exported files to the target system.

```
MySQL targethost:33060+ ssl JS > util.loadDump("~/dumps/hr-schema-tgt",  
{schema: "tgt-hr"})
```

6. When the data loading utility completes, open the @.json file in the source dump folder. This file contains the last committed transaction contained in the dump and a change data

Extract will need to be configured using that transaction value. Following is an example of the transaction information in the @.json file:

```
"binlogFile": "binlog.000005",
"binlogPosition": 1289,
  "gtidExecuted": "1174b383-3441-11e8-b90a-
c80aa9429920:1-9,\n1174b383-3441-11e8-b90a-c80aa9429921:1-9"
```

7. Add an Extract to the source database using either the gtidExecuted value, or the binlogFile and binlogPosition values, based on your database configuration.
 - a. If the source database is configured with gtid_mode set to ON, use the gtidExecuted value from the @.json file to add the Extract.

An example using GTIDSET positioning, based on the gtidExecuted value:

```
ADD EXTRACT extsrc, TRANLOG GTIDSET "1174b383-3441-11e8-b90a-
c80aa9429920:1-9,\n1174b383-3441-11e8-b90a-c80aa9429921:1-9"
```

- b. If the source database has gtid_mode set to OFF, use the binlogFile and binlogPosition values of the @.json file when adding the Extract.

An example using LOGNUM and LOGPOS positioning, based on the binlogFile and binlogPosition values:

```
ADD EXTRACT extsrc, TRANLOG, LOGNUM 5 LOGPOS 1289
```

8. Finally, complete the Oracle GoldenGate configuration by adding the following processes:
 - a. Corresponding trail for the Extract
 - b. A distribution path if needed
 - c. A Replicat to deliver to the target database
9. Start the Oracle GoldenGate processes and monitor the Extract and Replicat to confirm that they are running correctly, and that lag is reduced to near real-time.

Precise Instantiation for SQL Server

Starting with Oracle GoldenGate 23ai (23.8), precise instantiation can be orchestrated using the initial load Extract for SQL Server. The precise instantiation method allows for the initial load of data from a source database to a target database while the source database remains online for application updates. This method ensures precise positioning of the change data capture and delivery processes without having duplicate data in the target database, and without the need to use HANDLECOLLISIONS in the Replicat.

Precise instantiation for Oracle GoldenGate for SQL Server requires setting the source SQL Server database to allow snapshot isolation and uses the INITIALLOADOPTIONS USESNAPSHOT parameter within the initial load Extract. This method can be used to instantiate any supported target database from any supported SQL Server source database.

The following is a sample precise instantiation method using a schema dump from the source instance, then loaded to the target instance, followed by configuration of Oracle GoldenGate for change data replication:

1. ADD TRANDATA to the source database tables.

```
DBLOGIN USERIDALIAS src_alias
```

```
ADD TRANDATA dbo.*
```

2. After TRANDATA completes, verify that the database is enabled for snapshot isolation and if not, manually enable it.

```
ALTER DATABASE tpcc_source SET ALLOW_SNAPSHOT_ISOLATION ON;
```

3. For Microservices Architecture, add a Purge CDC Data task in the WebUI. See [Purge CDC Staging Data](#) and [Purge Process](#) to know more.

4. Create an Initial Load Extract, using the INITIALLOADOPTIONS USESNAPSHOT parameter.

```
ADD EXTRACT einit, SOURCEISTABLE
```

```
--Sample Initial Load Extract Parameter File:
```

```
EXTRACT einit
USERIDALIAS tpcc_source2017
INITIALLOADOPTIONS USESNAPSHOT
EXTFILE ei, MEGABYTES 500, PURGE
TABLE dbo.*;
```

5. Start the Initial Load Extract.

```
START EXTRACT einit
```

6. When the Initial Load Extract completes and stops, view the report file, looking for a similar message as below, which will be used when creating the CDC Extract.

```
VIEW REPORT einit
```

```
INFO      OGG-05381  A consistent point is established in database
'tpcc_source' at LSN 0x00000488:00002a98:0002.
      OGG-05379  Create or position a Change Data Capture Extract to LSN
0x00000488:00002a98:0002. Example: ADD EXTRACT <extract-name> TRANLOG LSN
      0x00000488:00002a98:0002 or ALTER EXTRACT <extract-name> LSN
0x00000488:00002a98:0002
```

7. The source database option to allow snapshot isolation can now be set back to OFF, if it was manually enabled per step 2.

```
ALTER DATABASE tpcc_source SET ALLOW_SNAPSHOT_ISOLATION OFF;
```

8. Create a CDC Extract using the LSN output from the initial load Extract's report file.

```
ADD EXTRACT extcdc TRANLOG LSN 0x00000488:00002a98:0002
```

```
ADD EXTTRAIL ea, EXTRACT extcdc
```

9. Create and start an initial load Replicat that reads from the trail created by the initial load Extract.

```
ADD REPLICAT rinit, PARALLEL, EXTTRAIL ei, CHECKPOINTTABLE ggadmin.ggcheck
```

```
START rinit
```

```
--Sample Initial Load Replicat Parameter file
REPLICAT rinit
USERIDALIAS tgt_alias
BATCHSQL
SPLIT_TRANS_RECS 100000
MAP dbo.*, TARGET dbo.*;
```

10. When the initial load Replicat ends processing, create and start a CDC Replicat that reads from the trail created by the CDC Extract.

```
ADD REPLICAT repcd, PARALLEL, EXTTRAIL ea, CHECKPOINTTABLE ggadmin.ggcheck
```

```
START EXTRACT extcdc
```

```
START REPLICAT repcd
```

Precise Instantiation for PostgreSQL Using Initial Load Extract

Data synchronization from a source PostgreSQL database to an Oracle GoldenGate target can be accomplished with the optional method of using precise instantiation. This method was introduced with Oracle GoldenGate 21c (21.8.0).

Precise instantiation has the advantage of not requiring any collision handling in the target Replicat. This is important for targets that do not support collision handling, such as flat files. This method uses a database snapshot to synchronize the output of the initial load Extract with the starting position of the Change Data Capture Extract. This snapshot is managed by the initial load Extract, so it is not possible for multiple initial load Extracts to use the same snapshot. Therefore, this method is not supported when using multiple initial load Extracts to parallelize the workload.

The following example uses the Admin Client within Microservices Architecture. It is assumed that you are familiar with Oracle GoldenGate and have setup the source and target databases correctly, with all required prerequisites. These steps require a minimum of Oracle GoldenGate 21c (21.8.0) or higher.

Perform the following steps to set up end-to-end initial load and synchronization processes using the precise instantiation method:

1. Register a Change Data Capture (CDC) Extract with the source PostgreSQL database.

```
DBLOGIN USERIDALIAS src_alias
REGISTER EXTRACT extecdc
```


In this example, extecdc is the Extract name. For Microservices Architecture, use DBLOGIN USERIDALIAS for database connection setup.

2. Create an initial load Extract.

```
ADD EXTRACT extinit, SOURCEISTABLE  
EDIT PARAMS extinit
```

The initial load Extract parameter file must contain the INITIALLOADOPTIONS USESNAPSHOT parameter. For example:

```
EXTRACT extinit  
INITIALLOADOPTIONS USESNAPSHOT  
SOURCEDB USERIDALIAS src_alias  
EXTFILE west/ei, MEGABYTES 500, PURGE  
TABLE public.*;
```

See INITIALLOADOPTIONS to learn about the usage of this parameter with the USESNAPSHOT option.

3. Start the initial load Extract.

```
START EXTRACT extinit
```

4. When the initial load Extract has completed and stopped, review its report file to determine the positioning LSN to be used by the CDC Extract.

For example, in the following output, the positioning LSN to be used by the CDC Extract will be '0/173F770'.

```
INFO    OGG-100001 A consistent point is  
established in database 'tpcc' using replication slot ogg_initx_1234 at  
LSN 0/173F770  
and snapshot name '00000003-00000026-1'.
```

```
INFO    OGG-100002 Create or position the CDC  
extract to LSN 0/173F770. Example: ADD EXTRACT <cdc-extract> TRANLOG LSN  
0/173F770  
or ALTER EXTRACT <cdc-extract> LSN 0/173F770.
```

5. Create and start an initial load Replicat that reads the trail from the initial load Extract.

```
DBLOGIN USERIDALIAS tgt_alias
```

```
ADD CHECKPOINTTABLE ggs.checkpoint
```

```
ADD REPLICAT repinit, EXTTRAIL west/ei, CHECKPOINTTABLE ggs.ggcheckpoint
```

```
START REPLICAT repinit
```

Here is an example of the initial load Replicat parameter file:

```
REPLICAT repinit  
TARGETDB USERIDALIAS tgt_alias  
BATCHSQL  
MAP public.*, TARGET public.*;
```

6. Add and start the CDC Extract (extecdc) using the consistent LSN value referred to in the initial load Extract report file.

```
ADD EXTRACT extecdc, TRANLOG, LSN 0/173F770
```

```
ADD EXTTRAIL ea, EXTRACT extecdc
```

```
START EXTRACT extecdc
```

Here is an example of a CDC Extract parameter file:

```
EXTRACT extecdc  
SOURCEDB USERIDALIAS src_alias  
EXTTRAIL west/ea  
TABLE public.*;
```

7. When the initial load Replicat has processed all the initial load records, add and start a CDC Replicat that reads the trail from the CDC Extract.

```
ADD REPLICAT repecdc, EXTTRAIL west/ea, CHECKPOINTTABLE ggs.ggcheckpoint
```

```
START REPLICAT repecdc
```

8. Monitor the lag in both the CDC Extract and the CDC Replicat, and when they are both close to zero seconds, then the data stream from source to target database should be close to real-time.

Precise Instantiation Between PostgreSQL Environments Using pg_dump Utility

When using Oracle GoldenGate to replicate between two or more PostgreSQL compatible databases, you can use the `pg_dump` (`pg_dumpall`)/`pg_restore` backup utility that is included with PostgreSQL. This will create a backup or snapshot of the source database, which will be restored on the target system.

See [pg_dump](#) ([pg_dumpall](#)) and [pg_restore](#) in PostgreSQL documentation.

The following steps describe how to use this backup in conjunction with Oracle GoldenGate to ensure a precise initial instantiation, which does not require the source to be taken offline, or the use of `HANDLECOLLISIONS`. These steps use the Admin Client, but you can perform the same steps from Oracle GoldenGate Microservices web interface and the Rest API service endpoints.

Before you begin these steps, make sure that you have completed preparing PostgreSQL for Oracle GoldenGate. See [Prepare PostgreSQL](#) section for details.

1. Connect to the deployment from Admin Client:

```
CONNECT https://source:svrvmgrport DEPLOYMENT deployment_name AS
deployment_user PASSWORD deployment_password
```

2. Connect to the source PostgreSQL database using the DBLOGIN command. If you have not set up a database connection to the source database, then you can set up using the steps from [Add Database Connections](#) or using the ALTER CREDENTIALSTORE command from the Admin Client.

```
DBLOGIN USERIDALIAS alias
```

For example, if the alias is ggeast, then the command would be:

```
DBLOGIN USERIDALIAS ggeast
```

3. Add a checkpoint table to the source database.

```
ADD CHECKPOINTTABLE owner.table_name
```

For example:

```
ADD CHECKPOINTTABLE ggs.checkpoint
```

4. Create your Extract parameter file to capture the tables you want to replicate, and register a new Extract with the source PostgreSQL database using the REGISTER EXTRACT command, but do not add the Extract.

```
REGISTER EXTRACT extract_name
```

For example:

```
REGISTER EXTRACT extcdc
```

5. Create a temporary replication slot that will be used to create a starting point for our Extract process and for positioning the PostgreSQL snapshot. This can be done using the [CREATE_REPLICATION_SLOT](#) command. Here are the steps to configure the temporary replication slot:

- a. Log into the PostgreSQL database using psql as a user with the *replication* role.
- b. Use the replication=*database* option. This is the same user that you created in [step 2](#).

For example:

```
$ psql "dbname=pgsource replication=database user=ggeast"
Password for user ggadmin: psql (16.3)
```

Type "help" for help.

```
pgsource=# CREATE_REPLICATION_SLOT tslot TEMPORARY LOGICAL
test_decoding EXPORT_SNAPSHOT;
```

```
slot_name | consistent_point | snapshot_name | output_plugin
-----+-----+-----+-----
```

```
tslot | 1/4232A6B0 | 00000007-00001142-1 | test_decoding
(1 row)
```

- c. Note down the `snapshot_name` and the `consistent_point`.
6. Create the snapshot using the `pg_dump` (or `pg_dumpall` for clustered databases), as shown in the following example:

Note

This command may take some time to execute. It is recommended to issue this command using a `nohup` option to avoid the command from stopping if the session closes.

```
$ nohup /u01/pgsql-16/bin/pg_dump -h pgsource -p 5432 -U ggeast -
snapshot=00000007-00001142-1 -F c -b -v -f latestdump.db pgsource &
```

The `pg_dump` / `pg_dumpall` uses the `snapshot_name` received from the output in [step 5c](#). In the given example, the `pg_dump` option assumes that the entire database will be replicated. However, if you are replicating a subset of the database, you can dump just those objects.

7. Add the Extract using the `ADD EXTRACT` command. The Extract should be positioned at the `consistent_point` tracked in [step 5c](#):

```
ADD EXTRACT extract_name, TRANLOG, LSN lsn_value
```

For example:

```
ADD EXTRACT extecdc, TRANLOG, LSN 1/42332A6B0
```

Positioning the Extract at this LSN value, ensure that the Extract will not capture any data that was included in the dump.

8. Add the Extract trail using the `ADD EXTTRAIL` command.

```
ADD EXTTRAIL trail_name, EXTRACT extract_name
```

For example:

```
ADD EXTTRAIL east/ea, EXTRACT extecdc
```

9. Start the Extract.

```
START EXTRACT extract_name
```

10. (Optional) If you need to configure a distribution path to send the trail files to another server, use the `ADD DISTPATH` command.

```
ADD DISTPATH name SOURCE source_uri TARGET target_uri
```

For example:

```
ADD DISTPATH dpecdc SOURCE trail://localhost:9002/services/v2/sources?  
trail=ea TARGET wss://localhost:9003/services/v2/targets?trail=ea
```

11. Restore the database on the target server using the `pg_restore` command. Just like in the `pg_dump`, if the session is lost, the restoration will fail, so it is recommended to use the `nohup` option. In this example, the following options were used:

```
$ nohup /u01/pgsql-16/bin/pg_restore -h pgtarget -p 6432 -U ggadmin -d  
pgtarget -v  
latestdump.db &
```

12. (Optional) If you created a distribution path to a Receiver service in another deployment, you may need to use the Admin Client `CONNECT` command to connect to target deployment.

```
CONNECT https://target:svmgrport DEPLOYMENT deployment_name AS  
deployment_user PASSWORD deployment_password
```

13. Connect to the target PostgreSQL database using the `DBLOGIN USERIDALIAS` command. If you have not set up a database connection to the source database, then you can set up using the steps from [Add Database Connections](#) or use the `ALTER CREDENTIALSTORE` command from the Admin Client.

14. Add a checkpoint table to the target database.

```
ADD CHECKPOINTTABLE ggs.chkpoint
```

15. Add a Replicat that will apply the changes to the target database after the restoration of the database is complete.

```
ADD REPLICAT replicat_name
```

For example:

```
ADD REPLICAT repecdc
```

16. Start the Replicat after the restoration has completed, as shown in the following example:

```
START REPLICAT repecdc
```

Monitoring and Controlling Processing After the Instantiation

After the target is instantiated and replication is in effect, you can control processes and view the overall health of the replication environment.

If you configured Replicat in integrated mode, you can use the `STATS REPLICAT` command to view statistics on the number of transactions that are applied in integrated mode as compared to those that are applied in direct apply mode.

```
STATS REPLICAT group
```

The output of this command shows the number of transactions applied, the number of transactions that were redirected to direct apply, and the direct transaction ratio, among other statistics. The statistics help you determine whether integrated Replicat is performing as intended. If the environment is satisfactory and there is a high ratio of direct apply operations, consider using nonintegrated Replicat. You can configure parallelism with nonintegrated Replicat.

Note

To ensure realistic statistics, view apply statistics only after you are certain that the Oracle GoldenGate environment is well established, that configuration errors are resolved, and that any anticipated processing errors are being handled properly.

To ensure that all processes are running properly and that errors are being handled according to your error handling rules, see [Error Management](#). Oracle GoldenGate provides commands and logs to view process status, lag, warnings, and other information.

Verifying Synchronization

To verify that the source and target data are synchronized, you can use the Oracle GoldenGate Veridata product or use your own scripts to select and compare source and target data.

Distribute

Learn about the Distribution Service, how to add a distribution path, how to add a target-initiated distribution paths, and managing distribution paths.

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

Source/Target	MA Non-Secure	MA Non-Secure with NGINX	MA Secure	Classic Architecture
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](#).

About Distribution Service Path

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:

- **USERIDALIAS 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.

Note

Distribution Path created using **Certificates** requires target deployment to have an x-Cert user configured that has the same username as **Subject DN** of Distribution Path Certificate. See [Create External Trusted RootCA and Distribution Client Certificates](#). Also see, [Connecting Two Deployments Using External RootCA Certificate](#).

- **OAuth target authentication:** In this case, the Oracle GoldenGate user authentication is outsourced to an OAuth service such as MS Azure EntraID, IDCS, OAM, and IAM. While IAM is a cloud-based identity provider, OAM is an on-premise identity provider.

Depending on the target deployment mode (secure or non-secure), you can choose the target authentication as follows:

- Secure: Using certificates, USERIDALIAS, or external Identity Provider using OAuth2.0
- Non-Secure: Using USERIDALIAS method.

If you use the USERIDALIAS, then consider the following guidelines:

- The target Oracle GoldenGate instance contains a user identified with a password.
- The source Oracle GoldenGate instance includes a USERIDALIAS (example: ggnet_alias) that matches the target's user credentials (example: user ggnet with password ***).
- When a Distribution Path is initiated, the USERIDALIAS information will be passed to the target. After the target verifies that the USERIDALIAS information matches the target user credentials, the authorization succeeds, and a network connection can be established.

Note

- The USERIDALIAS used for the Network connection between source and target Oracle GoldenGate instance is unrelated to any database.

Add a Distribution Path

A path is created to send the trail data over the network from the Extract to the Replicat, creating a trail path.

Note

A distribution path cannot be created from a non-secure deployment to a secure deployment. It can be created between two secure deployments or from a secure deployment to a non-secure deployment using the wss protocol.

Note

Before you begin adding a distribution path, make sure you establish a path connection. To set up a path connection, see [Add a Path Connection](#).

You can begin adding a distribution service path from the Microservices home page itself or use the left navigation pane to access the Distribution Service Paths page, as mentioned in the following steps:

1. Click **Distribution Service Paths** option from the left navigation pane to open the **Distribution Service Paths** page.
2. Click the plus (+) sign next to **Distribution Service Paths**. The Add Path wizard is displayed.
3. On the **Path Information** screen, enter the following details:

Path Information Screen	Description
Path Name	Select a name for the path.

Path Information Screen	Description
Description	Provide a description. For example, the name of the Extract and Replicat processes associated with the distribution path.

4. On the Source Options screen, enter the following details:

Source Options Screen	Description
Source Extract	Select the source Extract for which the trail distribution will be done by the Distribution Path.
Trail Name	Name of the source trail file.
Subdirectory	Path of the trail subdirectory.
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.
Encryption Profile	Select an encryption profile from the drop down list, if required.
Begin	Select the point from where you need to log data. You can select the following options from the drop-down list: - Now - Custom Time - Position Log
Source Log	Specify the values for the source log: - Sequence Number - RBA Offset

5. Click **Next**.
6. On the Target Options screen, specify the following details:

Target Options Screen	Description
Target Protocol	Select the data transfer protocol. Available options include wss, ws, and ogg.
Reverse Proxy Enabled	Select to use reverse proxy.
Deployment	Name of the target deployment.
URI Path	The URI path of the target host.
Target Host	Enter the URL of the target host, for example, localhost, if the target is on the same system.
Port Number	Port number of the target deployment.
Trail Name	Name of the source trail file.
Trail Subdirectory	Location of the trail subdirectory.
Trail Size	Maximum size of the file in the trail.
Target Encryption Algorithm	Encryption algorithm used when sending trail to target deployment. Options include AES128, AES192, and AES256.
Change Encryption	Enable this option to allow changing the encryption algorithm.

Target Options Screen	Description
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. This field is auto-populated.
Domain	The Domain that stores the userid alias, which has the target user id and password. This is used when the Distribution Path used the Target Authentication Method uses the UserID Alias authentication.
Alias	UserID alias that has the user credentials of the <i>source</i> Oracle GoldenGate deployment for a unidirectional replication or <i>target</i> for bidirectional replication. See About Distribution Service Path for details.
Trail Options	Enable this to customize the trail options in the next screen.
Advanced Options	Enable this to customize the other advanced options in the next screen.

7. Click Next.

8. On the Trail Options screen, enter the following details:

Trail Options Screen	Description
Type	Trail is accepted in the following formats: Plain Text XML SQL
Compatible With	Select the utility that is compatible with the trail file. Options are: • BCP • SQLLOADER • COMCAST
Timestamp Precision	Specify the timestamp precision value for the trail file. TS Date Time
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 SYSKEY.
Quote Style	Select the quote style depending on the database requirements. Options are: single none embed

Trail Options Screen	Description
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.
Include Operation Type	Enable this option to include the trail operation type in the trail options.
Include Image Indicator	Enable this option to include details about the before and after image.
Include Object Name	Enable this option to include the object name in the trail details.
Encoding	UTF-8
Use Qualified Name	Select to use the fully qualified name of the parameter file.
Include Transaction Info	Enable to include transaction information.

9. Click Next.

10. The Advanced Options screen appears if you enabled Advanced Options in the Target Options screen. On the Advanced Options screen, provide the following details to configure the target network:

Advanced Options	Description
Target Trail Sequence Length	The length of the trail sequence number.
Enable Network Compression	Set the compression threshold value if you decide enable this option.
Compression Threshold	Option appears when you enable the network compression. Specify the compression threshold value.
Eof Delay (centiseconds)	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.
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.

Advanced Options	Description
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.
Use Enckey	Enable this option to use an Enckey.
Target Encryption Keyname	The encryption keyname of the Enckey method.

11. Click **Next**.

12. On the Filtering Options screen, specify the following:

Filtering Options Screen	Description
Rule Name	Name of the rule you need to create.
Rule Action	Select from the Exclude or Include rules. Exclude would filter out based on the selected options, while Include would include data based on the specified options.
Filter Type	Select from the following list of options: - 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.
Object Names	Name of the object being filtered.
Negate	Select this check box if you need to negate any existing rule.

13. Click **Add** to add the rule and then click **Next**.
14. On the Managed Options screen, specify the following options:

Managed Options Screen	Description
Critical	Enable this option if you want to configure the Distribution Path for high availability.
Auto Restart	Enable this toggle switch to adjust the auto restart options: - Auto restart Retries - Auto restart Delay

15. Click **Create Path** or **Create Path and Run**, as required. Select **Cancel** if you need to get out of the Add Path page without adding a path.

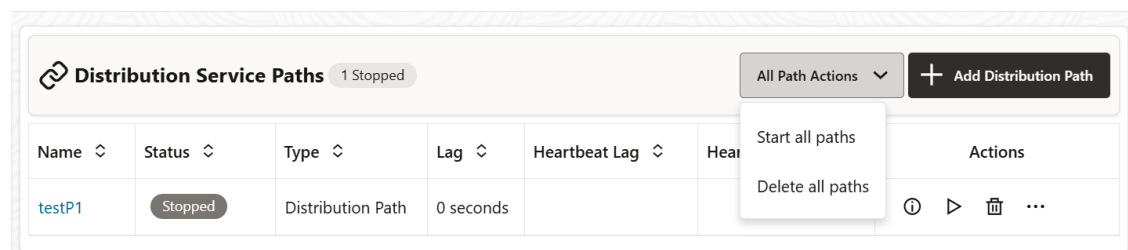
You are returned to the Distribution Service home page, which now includes your new path.

After the path is created, you'll be able to see the new path in the **Distribution Service** home page. You can also see this distribution path from the **Receiver Service** home page.

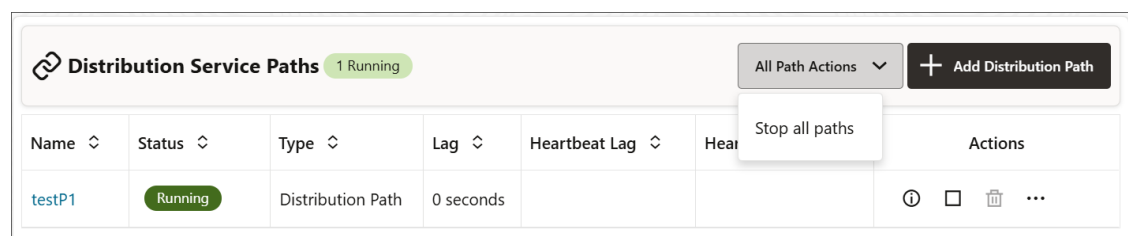
Actions for Distribution Service Paths

To manage distribution paths, you can perform various tasks such as stop or pause a path, view reports and statistics, reposition the path, change its filtering, and delete a path.

From the left-navigation pane, click Distribution Service Paths. The Distribution Service Paths page is displayed. To start and stop all distribution service paths associated with the deployment, you can select the Start All Paths or Stop All Paths options from the All Path Actions drop down list.



If the distribution service path is running, then the All Path Actions drop-down menu displays the option to **Stop all paths**.



From the distribution service paths list, you can use icons from the **Action** column to perform the following actions to manage the selected distribution path:

- **Details:** Use this icon 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 metrics for 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.
- **Start or Stop:** Use this icon options to start or stop a path. If the path isn't started, the Start option is displayed instead of the Stop option. For a target-initiated distribution path, you can only stop this path from the Distribution Service and cannot delete or start it from the Distribution Service. After you stop the path, it'll not be available on the Distribution Service.
- **Delete:** Use this icon option to delete a path. This option is available only when the path is in stopped state. Click Yes on the confirmation screen to complete path deletion.
- **Reposition:** Use this option to change the Source Sequence Number and Source RBA Offset. See [Reposition a Path](#).
- **Change Filtering:** Use this option to enter sharding, DML filtering, DDL filtering, Procedure filtering, and Tag filtering options. See [Change the Path Filtering](#).

Reposition a Path

You can reposition a path as required. To reposition a distribution path or target-initiated distribution path:

1. Click Distribution Service Paths from the left-navigation pane to to open the Distribution Service 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, click Distribution Service Paths to open the Distribution Service Paths page.

For the specific distribution path, click **Action**. From the drop-down list, click **Change Filtering**.

Rule Configuration	Task
Add paths	If you enable filtering by selecting it from the toggle button and click Add Rule, you'll see the Rule Definition dialog box. • Rule Name • Rule Action: Select either Exclude or Include • Filter Type: Select from the following list of options: – 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. You can also see the JSON script for the rule by clicking the JSON tab.

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	If you enable filtering by selecting it from the toggle button and click Add Rule, you'll see the Rule Definition dialog box. • Rule Name • Rule Action: Select either Exclude or Include • Filter Type: Select from the following list of options: – 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. Note Starting with Oracle GoldenGate 23ai, CDBs are only used with Downstream Extracts. – 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. You can also see the JSON script for the rule by clicking the JSON tab.

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*.

Monitor the Path Information and Statistics

Learn about monitoring the path information and statistics for distribution service paths and receiver service paths.

Monitor the Path

To review the path information, click the name of the path under **Distribution Service Path** from the left-navigation pane. You'll be able to view the following details about the receiver service path:

- **The Basic Path Information:** This includes the path status, source URI, lag time, target URI, source authentication method, source trail name, source trail file name, source sequence number, source RBA Offset, target encryption algorithm, and auto restart options.
- **Filter Rules:** This displays the applied filtering rules for the path.
- **Encryption:** This includes encryption method used for the path. Option can be Local Walle or the name of the encryption profile that is applied to the path.
- **Advanced Options:** This includes checkpoint frequency value, EOF delay, TCL Flush and other advanced options as configured during the creation of the path.

Review the Path Statistics

To review and edit the Distribution or Receiver Service Path statistics, click the name of the Distribution/Receiver Service Path from the left-navigation pane. You'll be able to view the following statistics related to the selected Distribution Service Path:

- **LCR Table:** This includes the type of LCR and the current value of the LCR.
- **DDL/DML Table:** This includes the type of transaction, DDL or DML and displays the number of Inserts, Updates, Upserts, and Deletes performed.
- **Statistics Table:** This includes statistical details of the path including the number of LCR read and LCR sent values. You can also search for specific entries from this table by using the Search box.
- **Network Statistics Table:** View the network metrics such as receive/send wait time, messages received/sent, and bytes received/sent.

About Receiver Service Paths

Receiver Service paths, also known as target-initiated distribution paths, initiates a path from the target deployment to the Distribution Service on the source and pull trail files.

The Receiver Service Paths create a target initiated path for environments such as Demilitarized Zone Paths (DMZ) or Cloud to on-premise, where the Distribution Service Paths in the source Oracle GoldenGate deployment cannot open network connections in the target environment to the Receiver Service Path due to network security policies.

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

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

Add Target-Initiated Distribution Paths

Target-initiated paths for Microservices enable the Receiver Service Path on the target to initiate a path connecting to the Distribution Service on the source deployment and pull trail files from the source.

To create a target-initiated distribution path:

1. From the left-navigation pane, select **Receiver Service Paths** to open the Receiver Service Paths page.
2. Click the **Add Target-Initiated Distribution Path** button on the top-right corner of the page to open the Add Path dialog box.
3. The following table lists the options to set up the path:

Table 8-1 Add a Target-Initiated Distribution Path

Screen and Options	Description
Path Information Screen	Provide a name and description for the target-initiated path on this screen
Path Name	Name of the target-initiated distribution path
Description	Provide a description of the path.
Source Options Screen	Specify the options for the source deployment which will be connected by the target-initiated path.
Reverse Proxy Enabled	Select to use reverse proxy. To know more about configuring your reverse proxy servers, see Reverse Proxy Support.
Source Authentication Method	Select the authentication method for the source URI. Authentication options are OAuth 2.0, Certificate, UserID Alias.

Table 8-1 (Cont.) Add a Target-Initiated Distribution Path

Screen and Options	Description
Source Protocol	From the drop-down list, select your data transfer protocol. The default option is Secure Web Socket Protocol (wss). Other option is ws .
Source Host: - Domain: Enter the domain for the host. - Alias: Provide an alias for this host. Path takes the source trail and sends the data to a target trail given here, which can be consumed by any Replicats created later.	Specify the URL of the source host. For example, localhost , if the source is on the same system.
Port Number	Enter the port number of the Distribution Service.
Trail Name	Enter the trail name you want to read on your source. NOTE: The Distribution Service doesn't not create any trail on source. It can only read the provided trail name.
Trail Subdirectory	Specify the trail subdirectory (if any).
Trail Size	Maximum size of the trail file that can be received from the source.
Generated Source URI	A URI is automatically generated for the trail based on the source information you provided.
Encryption Profile	Select the name of the encryption profile for receiving trail data. Also see Encrypting Trail Files .
Source Authentication Method	The source authentication method is preselected based on the selection made on the source deployment.
Domain	This option is preset to Network.
Alias	Provide an alias for the source host.
Begin	This option allows configuring the position from where the log data is received. The options are: - Now: The current position to begin received trail data. - Custom Time: Receive trail data between the specified time period. - Position in Log (default): Receive trail data based on the position in log. - CSN (at or after): Specify a CSN value at which or after which the trail data would be received.
Source Log: Sequence Number	Specify a sequence number value for the source log.
Source Log: RBA Offset	Specify RBA offset value for the source log.
Target Options	Specify the Trail file and directory for the target deployment.
Trail Name	Name of the target trail of the Replicat you created earlier.
Subdirectory	Name of the trail subdirectory.

Table 8-1 (Cont.) Add a Target-Initiated Distribution Path

Screen and Options	Description
Target Encryption Algorithm	Select the encryption algorithm for the target trail. Options include AES128, AES192, AES256.
Generated Target URI	A Target URI is automatically generated for the trail based on target trail information you provided. Click the pencil icon, if you want to edit it.
Change Encryption	To allow changing the encryption algorithm after the path is created, enable this toggle switch.
Target Type	Select one of these types of trail file formats: • GGFormat • Plain Text • XML • SQL
Advanced Options	Specify advanced network options.
Enable Network Compression	Enable this toggle switch to specify a compression threshold value.
Compression Threshold	Set the compression threshold value if you decide enable this option.
Sequence Length	The length of the trail sequence number.
Eof Delay (centiseconds)	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.
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.
Filtering Options	Specify rules for filtering within the trail.

Table 8-1 (Cont.) Add a Target-Initiated Distribution Path

Screen and Options	Description
Rule Name	Name of the rule you need to create.
Rule Action	Select from the Exclude or Include rules. Exclude would filter out based on the selected options, while Include would include data based on the specified options.
Filter Type	Select from the following list of options: • 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.
Object Names	Name of the object being filtered.
Negate	Select this check box if you need to negate any existing rule.
Managed Options	Additional options to configure auto restart.
Critical	Enable this toggle switch if configuring auto restart options based on high availability requirements.
Auto Restart	By default this toggle switch is enabled. If disabled, the path is not restarted automatically when killed.
Autorestart Retries	The number of times to try an restart the task (path process).
Autorestart Delay	The duration interval to wait between retries.

4. Click Create and Run to start the path.

For target-initiated distribution paths, the use case for the ws and wss protocols is explained in the following table:

Deployment Type	Target Deployment (Non Secure)	Target Deployment (Secure)
Source Deployment (Non-secure)	WS	WS
Source Deployment (Secure)	WSS	WSS

The wss protocol must be specified whenever the source deployment (Distribution Service 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 Service host) has disabled security.

Features and Limitations for Using Target-initiated Distribution Paths

Here are the limitations when working with target-initiated distribution paths:

- There is no support for interaction between legacy and secure deployments using this mode of operation for target-initiated distribution paths.
- No support for ogg protocol. Only ws and wss protocols are supported.
- It is possible to only get information and stop a target-initiated distribution path on Distribution Service and after the path stops, it is not be visible on the Distribution Service.

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`.

Actions for Target-Initiated Distribution Path

From the left-navigation pane, click Receiver Service Paths and select the path name. This page allows you to view all the target-initiated distribution paths that are in 'Running' state along with their status, lag time, and details about the target-initiated distribution path.

From the **Actions** column of the Target-Initiated Distribution Paths table, you can:

- Click the **Details** icon to view the path information. You will reach the **Path Information** page. You can also access the Path Information from the left navigation pane.
- Click the **Start/Stop** icon to start or stop the path.
- Click the **Delete** icon to delete a path. Click OK to confirm deleting the path.
- Click the **three-dots** icon to forcefully stop the path using the Force Stop option.

Access Network Statistics for a Path

From the Receiver Service Path option, you can monitor the network statistics and File input-output statistics of the paths. To view the network statistics of a path:

1. From the left navigation pane, click Receiver Service Paths and select Network Statistics.
2. Refresh the Network Statistics page if you don't see the latest information on the page.

The Network Statistics for distribution path include the following details:

- Target trail file name.
- Source host name for the distribution path.
- Transfer protocol for the distribution path. Options include ogg, wss, and ws.
- Port number of the Distribution Service.
- Total bytes received.
- Total bytes written out
- Total messages received
- Total messages written out
- Waiting time for writing messages
- Waiting time for receiving messages
- File IO Statistics including total bytes read, total bytes written to file, total idle time.

About Data Streams

Oracle GoldenGate Data Streams offers application developers and data scientists an approach to access real-time transactional data captured by Oracle GoldenGate. This technology streamlines data ingestion by eliminating the need for intermediary systems and facilitates seamless integration with existing workflows.

About Data Streams

Oracle GoldenGate Data Streams utilizes the AsyncAPI specification for defining asynchronous APIs. This approach enables applications to efficiently subscribe to data streams using a Publish or Subscribe model. Updates are received as soon as changes are committed in the source database, minimizing latency and simplifying application development. Additionally, Oracle GoldenGate Data Streams enable users to specify their preferred data format, such as JSON, for seamless integration with existing tools and frameworks within their development environment.

Benefits for Developers and Data Scientists

- **Enhanced Data Ingestion:** The Publish or Subscribe model powered by AsyncAPI enables applications to efficiently receive real-time data updates.
- **Flexible Data Formatting:** Users can choose their preferred format for seamless integration with existing tools.
- **Streamlined Integration:** AsyncAPI fosters smooth interaction with various applications and tools commonly used by developers and data scientists.
- **Guaranteed Data Integrity:** Inheriting the core strength of Oracle GoldenGate, Data Streams ensure data durability by replicating changes as they are committed in the source database.

Note

The Data Streams functionality is licensed with Oracle GoldenGate for Distributed Applications and Analytics or with Oracle GoldenGate for Big Data Targets. Refer to the Oracle GoldenGate for Distributed Applications and Analytics in the *Oracle GoldenGate Licensing Guide* for details.

Components of Oracle GoldenGate Data Streams

Components of Oracle GoldenGate Data Streams include:

- Async API
- Data Streams Protocol
- Data Streams Start/Restart Position
- Schema Records
- CloudEvents Format

Async API

Oracle GoldenGate Data Streams is programming language agnostic so that it can interact with a client written in any programming language. Even though the client programs typically are simple and small, users still need to manually implement the client code to interact with the data streaming service.

Adopting the AsyncAPI specification into Oracle GoldenGate Data Streams has the following advantages:

- Ability to describe the data streams service API in industry-standard API specification and automatically generate API documentation.
- Automatically generate client-side code with @asynccapi/generator.

With AsyncAPI support, Oracle GoldenGate Data Streams simplifies data streaming by generating the client code automatically. It follows the publisher and subscriber model and support a wide variety of protocols including websocket, kafka, mqtt, hms, and many IOT protocols. When describing an event-driven API, it uses the YAML modeling language and follow similar syntax for OpenAPI specification.

For example, the following snippet of AsyncAPI YAML document describes Data Streaming AsyncAPI definitions:

```
asynccapi: '3.0.0'
info:
  title: Data Streaming API
  version: '1.0.0'
  description: | allows clients to subscribe to a data stream
  license:
    name: Apache 2.0
url: 'https://www.apache.org/licenses/LICENSE-2.0'

servers:
  EAST:
    protocol: ws
url: east.oraclevcn.com:9002
```



```
defaultContentType: application/json
```

```
channels:
/services/v2/stream/mystream1:
...
```

When a data streams resource is created, a URL link to a customized Async API specification document describing how to access this data stream endpoint, is returned in the HTTP response. This YAML document can then be used to generate the client-side code using **@asynccapi/generator**.

Note that to support the websocket protocol in **@asynccapi/generator**, you also need to implement/maintain the websocket client template for the **@asynccapi/generator** in GitHub.

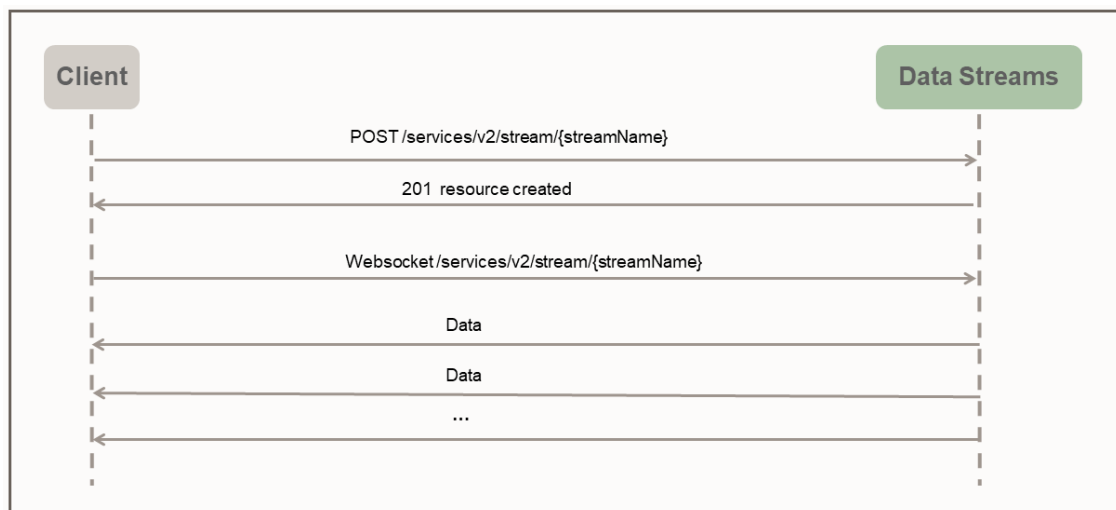
Refer to the GitHub repository for more information about the websocket-client-template:

<https://github.com/oracle-samples/websocket-client-template>

Data Streams Protocol

With Oracle GoldenGate Data Streams, direct access to the data in user specified format is enabled through a dedicated websocket channel that follows a simple streaming protocol.

Data Streams protocol uses push mode to send data to the client. The client first creates a streaming resource on the server through HTTP RESTful request. After the streaming resource is created, the client establishes a WebSocket connection through the streaming resource endpoint. After the WebSocket channel is established, Data Streams starts to push the data immediately and continuously without waiting for response or acknowledgement from the client.



The following sample python client illustrates the interaction between the client and the data streaming service:

```
import asyncio
import requests
import websockets
import json
```

```

async def client():
    ### create the streaming resource
    payload = {"source":{"trail":"a1"}}
    response = requests.post(
        'http://name:pswd@localhost:9002/services/v2/stream/s1', json=payload)

    ### establish websocket connection and receive data continuously
    uri = "ws://name:pswd@localhost:9002/services/v2/stream/s1?begin=earliest"
    async with websockets.connect(uri) as websocket:
        while True:
            resp = await websocket.recv()
            records = json.loads(resp)
            for rec in records:
                print(rec)
    asyncio.get_event_loop().run_until_complete(client())

```

In the given client program, a simple Data Stream payload specifying the source data trail name is provided when creating the data stream resource endpoint **s1**. In a real world application, much complicated Data Stream payloads can be used during the handshake phase of the streaming protocol to configure the data streaming behavior.

For example, the following Data Stream request payload specifies the filtering rules, encoding format, and bufferSize along with the required data source trail name.

```

{
  "$schema" : "ogg:dataStream",
  "source" : {"trail":"a1"},
  "rules" : [{
    "action" : "exclude",
    "filter" : {
      "objectNames" : [ "PDB2.AR.*", "PDB1.U1.*" ]
    }
  }],
  "encoding" : "json",
  "bufferSize" : 2097152
}

```

Data Streams Start/Restart Position

During the websocket connection establishment, client side specifies the **begin** position (as a query parameter in the websocket connection URL) to start streaming the data. The begin position can be one of the following values:

- Special keyword “*now*”
- Special keyword “*earliest*”
- ISO 8601 format timestamp string
- Last processed LCR position

Each non-metadata LCR record contains an opaque position (includes CSN, XID, record # inside the transaction). Client side is responsible for maintaining the position of the last processed LCR record. The data streams service is responsible for locating the correct start/restart point based on the given **begin** position.

If this is the first time a client connects to the data streams service, client should provide a timestamp of where to start streaming data. The keyword **now** will be converted to the current timestamp and the keyword **earliest** will be converted to the timestamp 0.

Alternatively, an ISO 8601 timestamp string can be used for **begin** position. In all cases, the data streams service performs a timestamp-based lookup on the source trail to determine the start position.

If this is the recovery/restart case, client should provide the saved last processed position to the data streams service during handshake. The data streams service will perform a position-based lookup on the source trail to determine the start position. The behavior of data streaming recovery also depends on the QoS level specified in the data stream.

Schema Records

Oracle GoldenGate Data Streams sends the following JSON Schema records to help clients to interpret the records in the data streaming service. There are four types of records in Data Streaming service:

- DDL operation record
- DML operation record
- Object Metadata record
- Data Streams Metadata record

Oracle GoldenGate Data Streams sends the corresponding schema record before sending any type of data or metadata records.

CloudEvents

CloudEvents is a specification for describing event data in common formats to provide interoperability across services, platforms and systems. As Oracle GoldenGate Data Streams currently only supports JSON data encoding, the support for CloudEvents format is limited to the JSON event format. The complete specification for JSON Event Format for CloudEvents can be found at:

<https://github.com/cloudevents/spec/blob/main/cloudevents/spec.md>

<https://github.com/cloudevents/spec/blob/main/cloudevents/formats/json-format.md>

<https://github.com/cloudevents/spec/blob/main/cloudevents/formats/cloudevents.json>

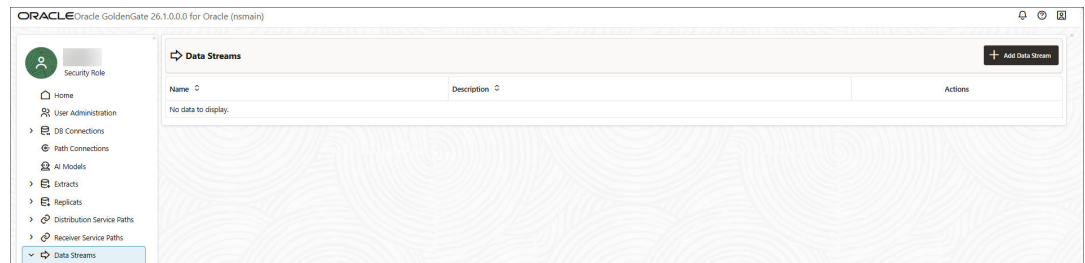
CloudEvents format defines the list of attributes to describe the event, essentially an envelope with a set of mandatory and optional attributes. When CloudEvents format is enabled in Oracle GoldenGate Data Streams, the final JSON records will look similar to the following, where **data** field contains the original data records, which is Oracle GoldenGate DML/DDDL/metadata/schema records.

```
{
  "specversion" : "1.0",
  "type" : "com.example.someevent",
  "source" : "/mycontext",
  "id" : "A234-1234-1234",
  "datacontenttype" : "application/json",
  "data: {...}
}
```

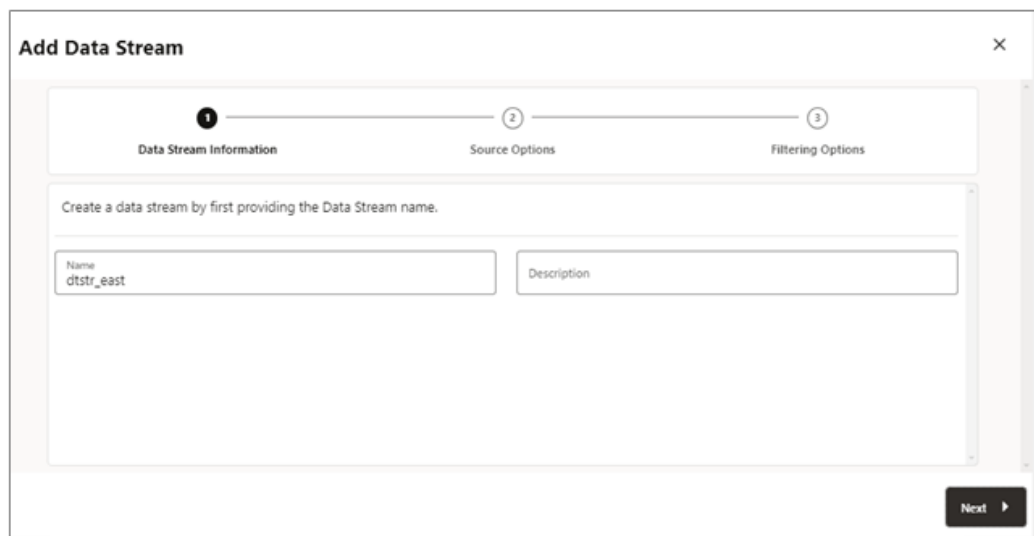
Add Data Streams

Here are the steps to create a Data Stream:

1. From the left-navigation pane, select **Data Streams** to open the Data Streams page.
2. Click the **Add Data Stream** button. The Add Data Stream dialog box is displayed.



3. On the **Data Stream Information** page, enter a Data Stream process name in the **Name** box and add a description for it. Click **Next**.



4. On the **Source Options** page, provide the values for options shown in the following image:

Add Data Stream

1 Data Stream Information 2 Source Options 3 Filtering Options

Choose the source trail file for the data stream, and the quality of service (exactly once, at least once, at most once). Optionally choose if you want the events delivered in the CloudEvents format.

Trail Name: e3

Trail Subdirectory: /east

Encoding: JSON

Buffer Size (bytes): 1,048,576

Quality Of Service: Exactly Once

CloudEvents Format: ☐ Data records conform to cloudEvents format

Back Next

- **Trail Name:** Name of the source trail file.
 - **Trail Subdirectory:** The path of the subdirectory where trail files are stored.
 - **Encoding:** This option controls the encoding of records for the data stream. Currently, JSON encoding is supported.
 - **Buffer Size:** This value controls the memory buffer size used in the data streaming service. The data streaming service will flush the in-memory message queue after its total byte size exceeds the specified value and delivers the records to the client.
 - **Quality of Service:** This option defines the data stream duplicate suppression and recovery behavior. Three levels of quality of service are supported in the data streaming service.
 - **Exactly Once:** This mode is the most restrictive mode where the service filters out any duplicate records during recovery of seeing a RESTART OK/ABEND records in the source trails. Clients will not see duplicate records. If the service cannot locate the record with given last processed position, an error occurs.
 - **At Least Once:** This mode does not suppress duplicate records during recovery or seeing a RESTART OK/ABEND records in the source trail. Clients may see duplicate records in the data stream. Also, if the service cannot locate the record with given last processed position, an error occurs.
 - **At Most Once:** This mode suppresses duplicate records during recovery or seeing a RESTART OK/ABEND records in the source trail. Clients will not see duplicate records in the data stream. If the service cannot locate the record with given last processed position, it will find the next available record and move forward.
 - **CloudEvents Format:** The data streaming service supports transmitting the data records in **CloudEvents** format. By default this format is disabled and can be controlled by enabling the property using the toggle switch, when creating the data streaming channel.
5. On the **Filtering Options** page, the options for include and exclude filtering rules are available, as shown in the following image:

Add Data Stream

Progress: 1. Data Stream Information (✓), 2. Source Options (✓), 3. Filtering Options (●)

Optionally add filtering rules (inclusion and exclusion rules) based on object names and object types.

Description:

Rule Action: ☒ Exclude ☐ Include

Filter Type:

Object Types: 1

- ☒ DML
- ☐ DDL
- ☐ INSERT
- ☐ UPDATE
- ☐ UPSERT
- ☐ DELETE

Specify the filtering rule options, as follows:

- **Rule Action:** Select Exclude or Include options.
- **Filter Type:** The filter type includes the following:

Object Type: You can select multiple options from the drop-down list including **DML**, **DDL**, **INSERT**, **UPDATE**, **UPSERT**, **DELETE**.

Object Names: Name of the previously created filtering object.

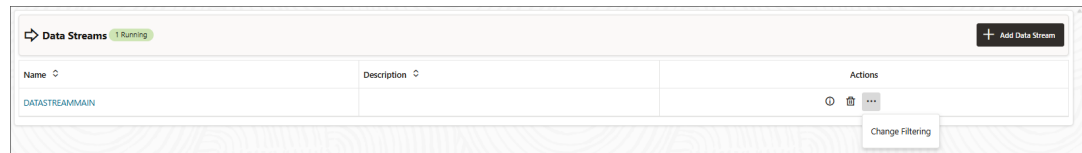
Click **Add** to add the filtering rule to the Data Stream process.

6. Click **Create Data Stream**. You will be returned to the Distribution Service home page where the Data Stream is listed.
7. Click on created Data Stream to view the YAML document Data Streaming AsyncAPI definitions for the stream.

Edit Data Streams Configuration

To edit the Data Streams configuration:

1. From the left navigation pane, click Data Streams.
2. Select the name of the data stream that needs to be modified.
3. From the Data Streams page, use the **Action** column to view data stream details, delete a data stream, and change its filtering.



4. If you click the **Change Filtering** option from the **Action** menu, the **Edit Data Stream Filtering Rules** dialog box is displayed. From this box, you can change the **Rule Action** (Include, Exclude) and **Filter Type** (Object Name or Object Type).
5. Click **Add** to apply the filtering.
6. Click **Submit** to return the Data Streams page.

You can edit the data stream further by clicking the data stream from the **Name** column. This displays the complete data stream configuration. Use the pencil icon next to each configuration setting, to change it. You can change the source trail file used by a data stream, it's filtering rules, and quality of service. You can also use the YAML editor to change the data stream configuration and upload changes using the **Upload Changes** icon next to YAML Editor.

Access and Edit Data Streams Path Information

To view or edit a data stream configuration:

1. From the left navigation pane, click **Data Streams** and select the data stream that you need to edit.
2. Click **Data Streams Information** option to open the current data stream configuration.
3. Click the pencil icon next to the **Data Stream Configuration** to edit the values for available trails, trail name, buffer size, Quality of Service, and the option to set if the data records conform to the **cloudEvent** format.
4. Click **Submit** to apply the changes.
5. Click the pencil icon for Filter Rules to change filtering option. The Edit Filter Rules dialog box is displayed.
6. Change the **Rule Action** (Include, Exclude) and **Filter Type**. See the section for details.
7. Click **Add** to apply the filtering.
8. Click **Submit** to return the Data Streams page.
9. You can edit the data stream further by using the YAML edit [How to Edit Filter Rules](#) or to change the data stream configuration and upload changes using the Upload Changes icon next to YAML Editor.

How to Edit Filter Rules

You can change a path filter using the following steps:

1. From the left-navigation pane, select the **Data Stream** and then select **Path Information**.
2. Select the **Filter Rules** section, click the pencil icon to open the Path Filtering Rules wizard.
3. Reset the filtering options as required. Available options are:

Filtering Options Screen	Description	
Rule Name	Name of the rule you need to create.	-
Rule Action	Select from the Exclude or Include rules. Exclude would filter out based on the selected options, while Include would include data based on the specified options.	-
Filter Type	Select from the following list of filter types and configure the related values:	Options for the Selected Filter Type
"	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. Select a Tag option: None, All, Customize. If you select to customize the tag, then you will be presented with options to enter a tag value, tag mask, and choose the values would be specified in binary or hex format.
"	Chunk ID	Displays the configuration details of database shards, however, the details can't be edited.

Filtering Options Screen	Description	
"	User Token	Select this option to apply user tokens for filtering. For this option, you need to provide the values for token name, token value, and then select a value match type, which can be set to Exact Match or Partial Match.
Negate	Select this check box if you need to negate any existing rule.	-

4. Click **Add**, and then click **Create** to complete adding the rule.

5. Click **Apply** to apply the changed rule to the path.

You can edit or delete rules as well. Any changes that you make must be applied to take effect.

Monitor Data Streams Statistics

To monitor the Data Streams Path statistics, click the name of the Distribution Service Path from the left-navigation pane and then select **Path Statistics**. You'll be able to view the following statistics related to the selected Distribution Path:

- **Statistics:** Get a snapshot view of the values for LCR sent/received, total DDLs, inserts, updates, and deletes, procedures, and any other value.
- **Table Statistics Table Search:** Search the table statistics table for specific values for Inserts, Updates, Deletes, number of LCRs read, and LCRs sent values.
- **Network Statistics:** View the network metrics such as receive/send wait time, messages received/sent, and bytes received/sent.

Oracle GoldenGate Data Streams REST APIs

You can use the following Rest APIs to manage GoldenGate Data Streams.

1. [Create a new Oracle GoldenGate Data Stream configuration](#)
2. [Retrieve an existing Oracle GoldenGate Data Stream configuration](#)
3. [Get a list of data stream resources](#)
4. [Update an existing Oracle GoldenGate Data Stream configuration](#)
5. [Retrieve the AsyncAPI YAML specification](#)
6. [Update the AsyncAPI YAML specification](#)
7. [Delete an existing Oracle GoldenGate Data Stream configuration](#)

Replicating Business Objects with Oracle JSON Relational Duality and GoldenGate Data Streams

Oracle GoldenGate 23ai has introduced replication of full-featured business objects in an event-based, pub/sub architecture. This is implemented by combining two new Oracle technology features: Oracle Database JSON Relational Duality Views and Oracle GoldenGate Data Streams.

Oracle AI Database 26ai JSON Relational Duality Views enables developers to easily consume relational data as JSON document models. This combines the advantages of JSON documents with those of the relational model, while avoiding the limitations of each. Oracle JSON relational duality view is essentially a SQL view that presents JSON data in a structured, relational format.

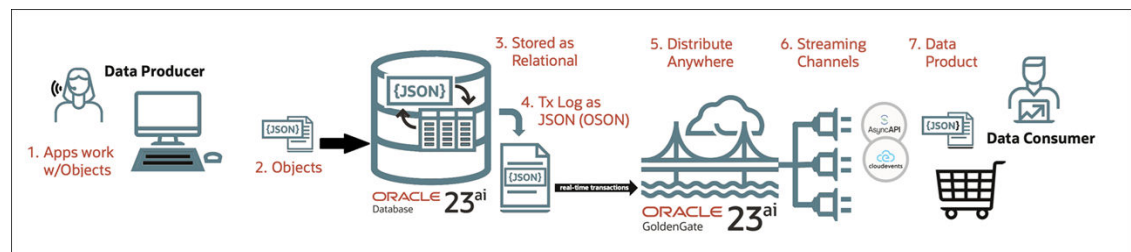
Replicating business objects with Oracle JSON Relational Duality and GoldenGate Data Streams prioritizes data products and becomes the single source of truth for applications, transaction fabrics, data/event meshes, and data fabrics by ensuring the uniqueness of data objects.

This topic covers the benefits of using Oracle JSON Relational Duality Views with GoldenGate Data Streams and provides the steps to configure GoldenGate Data Streams with Oracle JSON Relational Duality Views.

Advantages of Using JSON Duality Views with GoldenGate Data Streams

Replicating Business Objects with Oracle JSON Relational Duality and GoldenGate Data Streams prioritizes data products and becomes the single source of truth for applications, transaction fabrics, data/event meshes, and data fabrics by ensuring the uniqueness of the data objects.

The following workflow diagram illustrates the architecture when using JSON Duality Views with Oracle GoldenGate Data Streams:



The following are the strategic advantages of using this model:

- Overcoming impedance mismatch:** In JSON Duality Views, data is stored only once, underneath the normalized tables (preserving atomicity), while application objects work with JSON data objects. This way, business objects manage data products in Oracle JSON Duality Views in forms of JSON objects. Oracle GoldenGate captures JSON based data objects from JSON Duality Views and publishes events to AsyncAPI channels to make it available for subscribing data consumers. Integration between JSON Duality Views and Oracle GoldenGate addresses the impedance mismatch by guaranteeing exact once availability of the data products for both data producers and data consumers.
- Scalable Data Objects:** JSON Duality Views provide efficient storage management and data consistency as your application can work natively with JSON docs or with SQL on tables, or with both at the same time. Duality Views can be declared over any number of tables, plus even different duality views can be defined on the same or overlapping set of relational tables. This allows application developers to manage application objects in ways they are familiar and to create complex data products while using their usual drivers, frameworks, tools, and development methods, all while data is stored in a relational database.
- Atomic updates to event brokers:** JSON Duality delivers faster JSON than document databases and delivers better latency for OLTP by providing parallelism and optimization, as data management is done at the relational level with very high efficiency. Oracle

GoldenGate Data Streams automates the event-based publication of business objects directly from the data tier, in real-time, as the COMMIT events happen, using standard patterns.

- **ACID compliant data objects with CDC:** In JSON Duality Views, Lock-Free Concurrency Control keeps data objects constantly in-sync, while preventing both inconsistencies and stale data. Oracle GoldenGate Data Streams publishes trusted JSON-based data products with high throughput, reduced latency and guaranteed exact once availability of the data product for all data consumers.

Parameters Used When Configuring JSON DVs and JCTs

The following parameters need to be configured, when using JSON Relational Duality Views and JSON Collection Tables.

TABLE

Used by Extract to capture changes from JSON Relational Duality Views and JSON Collection Tables after logical replication is enabled. Wildcards for TABLE/MAP statements are supported.

Note

Although wildcards are supported in TABLE/MAP statements, ADD TRANDATA does not support use of wildcards.

The TABLE HR.* parameter setting will include base tables and JSON Relational Duality Views under HR if they are already captured by Extract, to avoid user overhead.

See TABLE | MAP in *Parameters and Functions Reference for Oracle GoldenGate*

MAP

Used by Replicat to apply changes to target JSON Relational Duality Views and JSON Collection Tables. Wildcards are supported. Ensure that the configuration doesn't apply changes to both the JSON DVs and its underlying tables to avoid duplication.

KEYCOLS

Not supported for JSON Relational Duality Views and JSON Collection Tables because they have a single, predefined key, _id.

AUTO_CAPTURE

Captures changes from JSON Relational Duality Views and JSON Collection Tables with enabled logical replication. See [Configure the Auto Capture Mode for Extract](#) and TRANLOGOPTIONS auto_capture mode for details.

Configure GoldenGate Data Streams and JSON Relational Duality Views to Deliver Change Data

When you combine the functionality of Data Streams with JSON Relational Duality Views to provide data delivery from the data producer to the data consumer, it requires setting up the JSON Relational Duality Views from Oracle AI Database 26ai and then creating Extract, the associated trail file, and adding Data Streams from the Oracle GoldenGate web interface.

For replicating into JSON Relational Duality Views and JSON Collection Tables, Oracle GoldenGate for Distributed Applications and Analytics MongoDB handler can be used. See MongoDB Handler in the *Oracle GoldenGate for Distributed Applications and Analytics* guide.

The steps to perform these tasks are as follows:

1. Create JSON Relational Duality Views.

For steps to create the JSON Relational Duality Views, see *Creating Duality Views in the JSON-Relational Duality Developer's Guide*.

2. Enable table-level supplemental logging for JSON Relational Duality Views and/or JSON Collection Tables. See [Enable Supplemental Logging for JSON Relational Duality Views and JSON Collection Tables](#).
3. Create an Extract and trail file in Oracle GoldenGate.
See [Add an Online Extract](#) and [Add a Trail](#).
4. Add a Data Stream from Oracle GoldenGate Distribution Service.
See [Add Data Streams](#).
5. Insert change data to the document for testing.
6. Consume the Change Data from the Data Stream.

For a sample set of cURL commands that would set up GoldenGate Data Streams for Duality Views, see [Sample Commands to Configure GoldenGate Data Streams for JSON Relational Duality Views](#).

Replicat

Learn about the Replicat process, its types, and steps to add a replicat, and other tasks associated with Replicat.

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 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 [Precise Instantiation for Oracle Using Initial Load Extract](#) 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.

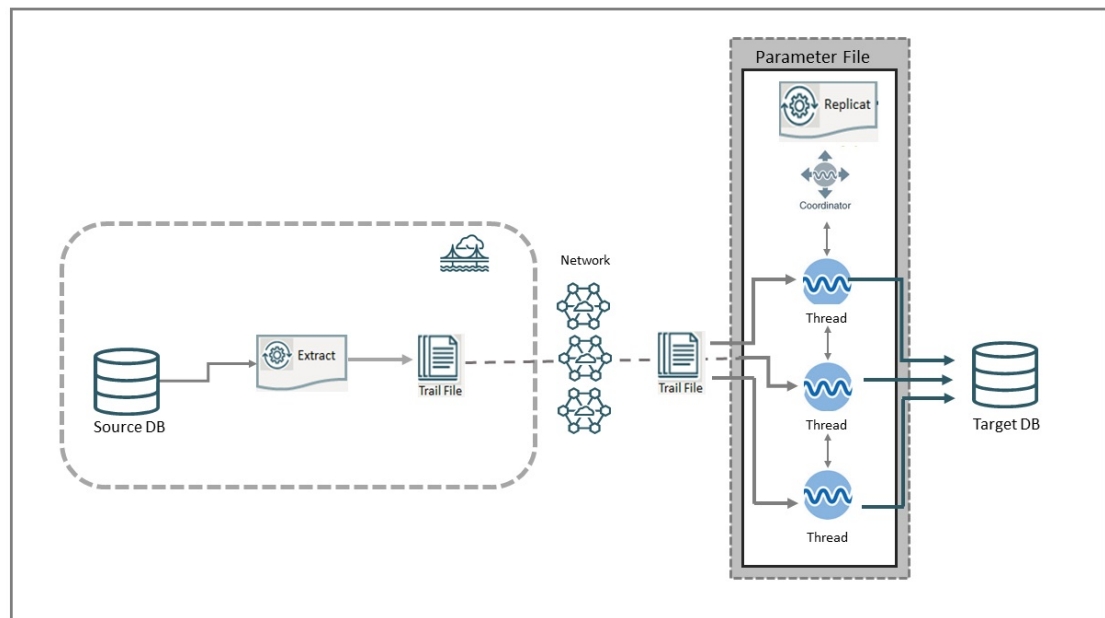
The difference between classic mode and coordinated mode is that Replicat is multi-threaded in coordinated mode. Within a single Replicat instance, multiple threads read the trail independently and apply transactions in parallel. Each thread handles the filtering, mapping, conversion, SQL construction, 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.

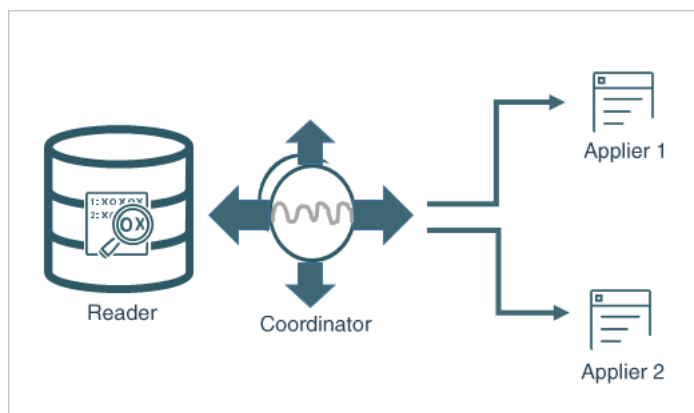
Note

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 [Transaction Log Settings and Requirements](#). This logging can be enabled at any time up to, but before you start the Oracle GoldenGate processes.
- Parallel Replicat in integrated mode 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 in integrated mode.

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

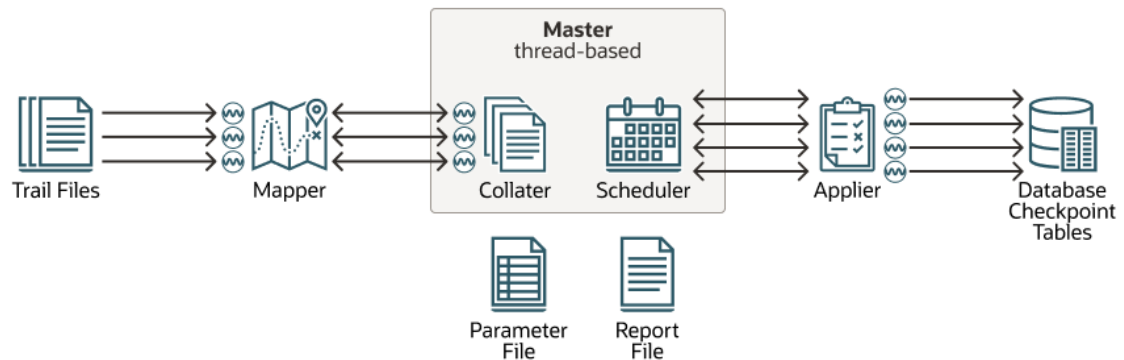
The following are the benefits of using parallel Replicat:

- 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	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.
MAX_APPLY_PARALLELISM	

Parameter	Description
SPLIT_TRANS_RECS	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
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

❶ Note

Auto-parallelism is disabled by default. Only your

Feature	Parallel Replicat	Integrated Replicat	Coordinated Replicat	Classic Replicat
	ead ad 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 p l i c			

Feature	Parallel Replicat	Integrated Replicat	Coordinated Replicat	Classic Replicat
	at 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 o - p a r			

Feature	Parallel Replicat	Integrated Replicat	Coordinated Replicat	Classic Replicat
	all elements must be used.			
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

Before you add a Replicat, make sure that the following settings are configured:

- [Add Database Connections](#)
- [Add TRANDATA](#)
- [Add a Checkpoint Table](#)
- [Add Heartbeat Table](#)

You can add Replicats for the target deployment from the Microservices portal page. After you've set up your source and target deployments, you can create and run the Replicat by following these steps:

1. From the Microservices home page, click the + (plus) sign next to **Replicats** to start the Add Replicat wizard.
2. Select a Replicat type from the **Replicat Information** screen.

Note

Some Replicat types are only available for certain databases. All Replicat types may not be applicable to your database.

The types of Replicat are:

- **Parallel Replicat:** If you select this option, then select an integrated or non-integrated parallel Replicat.
 - **Integrated:** This option appears when you select Parallel Replicat.
 - **Non-Integrated:** This option appears when you select Parallel Replicat.

Note

When configuring parallel Replicat for Sybase, make sure that you upgrade to version 16.0 SP3 PL15 and higher to prevent the Administration Service from disconnecting intermittently.

- **Integrated Replicat**
- **Non-integrated Replicat:** This option is displayed with heterogeneous or non-Oracle databases.
- **Coordinated Replicat**

Select the type of Replicat and specify the following:

- **Process Name:** The name of the Replicat process. For coordinated and parallel Replicats, the limit is 5 characters.
 - **Description:** Describes the Replicat process, text that can be entered and is longer than 5 or 8 characters to identify this process.
3. Click **Next**. The **Replicat Options** screen is displayed.
 4. On the **Replicat Options** screen, specify the following:

Replicat Options Screen	Description
Trail Name	A two character trail name.
Trail Subdirectory	Where you want the trail information to be stored or the name of the log file. The default trail file location is the Oracle GoldenGate Data Home.
Encryption Profile	List of the available encryption profiles. If you have not created an encryption profile, then the Local Wallet profile would be selected, by default. To know more about creating and applying encryption profiles, see Configure an Encryption Profile .
Credential Domain	Select a domain for the target endpoint.

Replicat Options Screen	Description
Alias	Select an alias for the target database.
Checkpoint Table	Set the use of an existing checkpoint table.
Begin	From where you want the Replicat to start. The following three options are available: • Position in Log • Now • Custom Time Now is the default option.
Custom Time	The Date and Timezone options appear when you select the Custom Time option from the Begin drop-down list. Specify the date and timezone as per your requirement, for Replicat to start.
Transaction Log Sequence Number	Set the log file number of the log file that contains the START TRANSACTION record for the transaction you are creating.
Transaction Log RBA Offset	Set the record offset of the log file you specified.
Max Threads Number	Valid for Coordinated Replicat. Set a hard limit for the maximum number of active threads that can run simultaneously or use the 25 threads default.

5. Click **Next**. The **Managed Options** screen is displayed.
6. On the Managed Options screen, specify the following 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.

 Note

This option only appears while creating the Extract or Replicat and not when you set up the managed processes in the Profiles page.

Auto Start	Enables autostart for the process
Startup Delay	Time to wait in seconds before starting the process
Auto Restart	Configures how to restart the process if it terminates

Option	Description
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.

7. Click **Next**. The **Parameter File** screen is displayed.
8. On the **Parameters File** screen, enter the basic parameters for setting up a Replicat. By default, the Parameter File text contains some parameters, which can be customized as required.

```

REPLICAT repe
USERIDALIAS ggwest DOMAIN OracleGoldenGate
DDL INCLUDE MAPPED
DDLERROR default discard
REPERROR (default,discard)
DDLOPTIONS REPORT
SOURCECATALOG DBEAST
MAP hr.*, TARGET hr.*;

```

9. Click **Create and Run** to start the Replicat after adding it. Click **Create** to simply add the Replicat but not start it immediately after being created.

Configure Replicat for Oracle Autonomous AI Database

This section assumes that the source environment is already configured and provides the steps required to establish replication in the Oracle Autonomous AI Database environment.

In the Oracle GoldenGate instance, you need to complete the following:

1. Follow the steps given in [Prerequisites for Applying to Oracle Autonomous Database](#).
2. Follow the steps given in [Unlock the Autonomous AI Database User for Oracle GoldenGate](#).
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 Autonomous AI 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 AI 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@database_name_low PASSWORD complex_password alias adb_alias
```

13. Add and configure a Replicat to deliver to Autonomous AI 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](#).

The following example creates a Replicat (required to replicate to an Autonomous AI Database) called rauto, and instructs it to begin now.

```
ADD REPLICAT rauto, PARALLEL INTEGRATED, EXTTRAIL ./dirdat/et
```

If you want to replicate HR.EMP into COUNTRY.EMPLOYEE, then your map statement would look like this:

```
MAP HR.EMP, TARGET COUNTRY.EMPLOYEE;
```

Note

You can use classic Replicat, coordinated Replicat, and parallel Replicat in non-integrated mode. Parallel Replicat in integrated mode is also supported for Autonomous AI Database.

14. You can now start your Replicat and perform data replication to the Autonomous AI Database. Here's an example:

```
START REPLICAT rauto
```

Note

Autonomous AI 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 to avoid having to manually restart a Replicat when it times out.

Replicat Actions

Replicat actions include starting or stopping Replicat, alter Replicat parameters, forcibly stopping Replicat, or deleting a Replicat.

Access Replicat Details

Process Information

Displays Replicat process details such as status of Replicat as running or stopped. You can also edit the encryption profile and managed options for auto start and auto restart from here.

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.

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 edit the parameters by clicking the edit **pencil** icon. See [Basic Parameters for Parallel Replicat](#). Also see [Additional Parameter Options for Integrated Replicat and Parallel Replicat in Integrated Mode](#)

Report

Displays the details about the Replicat including the parameters with which the replicat is running, and run time messages.

Stop, Start a Replicat

There are various options to start or stop a Replicat.

- **Start/Stop**: Select this option to start or stop a Replicat immediately.
- **Start/Stop (in the background)**: Select this option to start or stop Replicat using a background process.
- **Start with Options**: Select this option to change the Replicat start point, CSN value, filter duplicates before starting the Replicat. When you select this option, a screen is displayed where you can reset the CSN value, Replicat start point, and filter duplicates if required.
- **Force Stop**: Select this option to forcibly stop a Replicat process or group, immediately.

Alter Replicat

The Alter Replicat option allows you to reset some Replicat options. When you click Alter Replicat, a screen is displayed for options that you can modify for the specific Replicat or Replicat group. These options are:

- Replicat begin position.
- Replicat description.
- Intent of the Replicat.

Click **Submit** to apply the Replicat alterations.

Start the Replicat.

Delete Replicat

Select this action from the Replicat **Actions** button, when you have stopped the Replicat. This option allows you to stop Replicat processes in a group in the background.

Configure a Replicat Parameter File

Configure a Replicat parameter file to configure Replicat against a pluggable database. Replicat can operate in any mode within a pluggable database. These steps configure the Replicat parameter file.

1. On the target system, create the Replicat parameter file using Oracle GoldenGate command line interface.

```
EDIT PARAMS groupname
```

Where: name is the name of the Replicat group.

2. Enter the Replicat parameters in the order shown, starting a new line for each parameter statement.

Here is the sample Replicat parameter file:

```
REPLICAT repe
USERIDALIAS ggeast DOMAIN OracleGoldenGate

-- In case of Parallel Replicat:
MAP_PARALLELISM 2
APPLY_PARALLELISM 4
SPLIT TRANS_RECS 50000

-- In case of Integrated Replicat:
DBOPTIONS INTEGRATEDPARAMS(PARALLELISM 4)
DBOPTIONS INTEGRATEDPARAMS(EAGER_SIZE 20000)

DDL EXCLUDE ALL
DDLERROR default discard
REPERROR (default,discard)
DDLOPTIONS REPORT
SOURCECATALOG DBEAST MAP hr.*, TARGET hr.*;
```

Parameter	Description
REPLICAT <i>group</i>	group is the name of the Replicat group.
USERIDALIAS <i>alias</i>	Specifies the alias of the database login credential of the user that is assigned to Replicat. This credential must exist in the Oracle GoldenGate credential store.
DBOPTIONS INTEGRATEDPARAMS (parameter[, ...])	This parameter specification applies to Replicat in integrated mode. It specifies optional parameters for the inbound server.

Parameter	Description
MAP [container.]schema.object library/file library/file(member), TARGET [container.]schema.object library/file library/file(member);	Specifies the relationship between a source table or sequence, or multiple objects, and the corresponding target object or objects. - MAP specifies the source table or sequence, or a wildcarded set of objects. - TARGET specifies the target table or sequence or a wildcarded set of objects. - container is the name of a container, if the source database is a multitenant container database. - schema is the schema name or a wildcarded set of schemas. - object is the name of a table or sequence, or a wildcarded set of objects. - library is the IBM i library name or a wildcarded set of libraries. - file is the IBM i physical file name or a wildcarded set of physical files. - member is the IBM i physical file member name or a wildcarded set of member names. When using the IBM i native name format (library/file with optional member) the only valid wildcards are a name with at least one valid character followed by a trailing asterisk (*) or *ALL which matches any name.

Note

The member name is optional, and must be provided if the member names are required to be written in the trail as part of the object name. Without member names all members in a physical file be implicitly merged as a single object in the trail.

Terminate this parameter statement with a semi-colon.

To exclude objects from a wildcard specification, use the MAPEXCLUDE parameter.

- Enter any optional Replicat parameters that are recommended for your configuration. You can edit this file at any point before starting processing by using the EDIT PARAMS command.
- Save and close the file.

Additional Parameter Options for Integrated Replicat and Parallel Replicat in Integrated Mode

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.

The default Replicat configuration should be sufficient. However, if needed, you can set the following inbound server parameters to support specific requirements.

① 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 DBOPTIONS for more information about the 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_PGA_SIZE**: Controls the amount of Program Global Area (PGA) allocation on the target database. The default value for Oracle GoldenGate is 1 GB. If this parameter is not set, then it is bounded by **PGA_AGGREGATE_TARGET**.

- **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).
- **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).

Controlling Checkpoint Retention

The CHECKPOINTRETENTIONTIME option of the TRANLOGOPTIONS parameter controls the number of days that Replicat retains checkpoints before purging them automatically. Partial days can be specified using decimal values. For example, 8.25 specifies 8 days and 6 hours. The default is seven days.

Excluding Replicat Transactions in Bidirectional Replication

In a bidirectional configuration, Replicat must be configured to mark its transactions, and Extract must be configured to exclude Replicat transactions so that they do not propagate back to their source.

This can be implemented in two ways:

Method 1

Valid only for Oracle to Oracle implementations.

Replicat can be in either integrated or nonintegrated mode. Use the following parameters:

- Use **DBOPTIONS** with the **SETTAG** option in the Replicat parameter file. The inbound server tags the transactions of that Replicat with the specified value, which identifies those transactions in the redo stream. The default value for SETTAG is 00.
- Use the **TRANLOGOPTIONS** parameter with the **EXCLUDETAG** option in an Extract parameter file. The logging server associated with that Extract excludes redo that is tagged with the

SETTAG value. Multiple EXCLUDETAG statements can be used to exclude different tag values, if desired.

Method 2

Valid for Oracle as source.

Use the Extract TRANLOGOPTIONS parameter with the EXCLUDEUSER or EXCLUDEUSERID option to ignore the Replicat DDL and DML transactions based on its user name or ID. Multiple EXCLUDEUSER statements can be used.

Method 3

Valid for Non-Oracle sources.

This method requires TRANLOGOPTIONS FILTERTABLE to be used, for databases that support it, which are SQL Server, MySQL, and PostgreSQL.

DDL Notification on Target Tables

Oracle GoldenGate Replicat caches target table metadata during DML replication. Out of band DDL changes makes cached metadata obsolete. For example, when an application patch is applied to the target database before upgrading the source.

Starting with Oracle GoldenGate 23ai, Replicat uses efficient table DDL change notification to automatically detect target table metadata changes. Oracle GoldenGate registers for DDL notification on a table when the metadata is cached. Whenever DDL is performed on the target table, Replicat is notified and it invalidates any cached metadata. Replicat reloads the target table metadata, next time it applies DML to the target table. This feature is applicable to all types of Replicat for the Oracle Database. See TARGETDDLNOTIFY in the *Parameters and Functions Reference for Oracle GoldenGate* to learn about the parameter options and syntax.

For details about Efficient Table Change DDL Notification, see Using Efficient Table DDL Change Notification.

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.

MySQL: DDL Replication

Learn about DDL replication in MySQL.

Limitation of DDL Replication in MySQL

- **Case 1:** If DDL or the TRACKSCHEMACHANGES is enabled , and the table is in the capture list. Extract abends with the following error:

```
DDL statement "CREATE TABLE AS SELECT" is detected for table 'table-name'
at binary log
number 'binlog number' and offset 'binlog offset'. The DDL operation
"CREATE TABLE AS
SELECT" is not supported. Execute all 'CREATE TABLE AS SELECT' DDL
operations manually to the target
database and start Extract by adding Extract parameter TRANLOGOPTIONS
WARNCREATEASSELECT.
Once Extract gets past the 'CREATE TABLE AS SELECT' operation, restart
Extract after disabling
the parameter TRANLOGOPTIONS WARNCREATEASSELECT.
```

You need to take actions as suggested in the error message.

When the parameter TRANLOGOPTIONS WARNCREATEASSELECT is used in the Extract parameter file, Extract continues after issuing the following warning message in the Extract report, in case of subsequent 'CREATE TABLE AS SELECT' operation is found.

```
"DDL statement 'CREATE TABLE AS SELECT' is detected for table 'table-name'
at binary log
number 'binlog number' and offset 'binlog offset'. The DDL operation
'CREATE TABLE AS SELECT'
is not supported. The DDL and the associated DMLs are ignored."
```

- **Case 2:** When DDL replication or TRACKSCHEMACHANGES is disabled and the table is in the Extract list.

When both DDL replication / TRACKSCHEMACHANGES are not enabled in the capture, the DDL and associated DML operations with CREATE TABLE AS SELECT are ignored. User needs to issue the same "CREATE TABLE AS SELECT" in the target as it was issued in the source.

- **Case 3:** When the table is not in the capture list, then irrespective of the DDL or TRACKSCHEMACHANGES enabled or not, all the table operations are ignored.

Note

See TRANLOGOPTIONS WARNCREATEASSELECT in the *Parameters and Functions Reference for Oracle GoldenGate* for details.

MySQL: Prerequisites for DDL Configuration

The Plugin based DDL replication solution is not supported in Oracle GoldenGate 23ai and higher releases. The prerequisites for configuring transaction log based DDL replication are as follows:

- For the Transaction Log-based DDL replication, the extended metadata of the table is required in the binary logs, which is made available in the binlog by setting MySQL server's binlog_row_metadata variable to FULL. Therefore, it is mandatory to set binlog_row_metadata to FULL.

Note

The extended metadata in the binlog is available from MySQL Server 8.0.14 and MariaDB 10.5 onwards. As a result DDL replication previous to these versions is not supported.

1. Set the value of MySQL server variable binlog_row_metadata to FULL, inside MySQL configuration file, my.cnf for Linux and my.ini for Windows.
 2. Restart the database server after changing the configuration file for the settings to take effect.
- DDL replication for remote capture is supported for MySQL 8.0 onwards. Transaction log based DDL replication works with both remote or local capture. This was a limitation for earlier Oracle GoldenGate releases. For example, Oracle GoldenGate 19c remote capture did not support DDL replication.
 - Transaction log based DDL replication can handle DDLs issued within stored procedures, which was a limitation with plugin-based DDL replication.
 - By design, the heartbeat table DDLs are ignored by the capture and you should create the heartbeat tables manually at the target.

DDL Filtering for MySQL

The following options are supported for DDL filtering for MySQL DDL replication:

Option	Description
DDL INCLUDE OPTYPE CREATE OBJTYPE TABLE;	Include create table.
DDL INCLUDE OBJNAME ggvam.* EXCLUDE OBJNAME ggvam.emp*;	Exclude all the tables under the ggvam database with the emp wildcard.
 Note The EXCLUDE option needs to be added together with the INCLUDE option in this parameter.	
DDL INCLUDE INSTR 'XYZ' EXCLUDE INSTR 'XYZAB'	Include DDL that contains the string XYZ.
DDL INCLUDE INSTR 'WHYAB' DDL EXCLUDE INSTR 'WHY'	Excludes DDL that contains the string WHY.
DDL INCLUDE MAPPED	This is the recommended DDL option for MySQL DDL replication.
DDL EXCLUDE ALL	Default option.

 Note

EXCLUDE must always have INCLUDE statement except for EXCLUDE ALL else Extract will abend.

For a full list of options, see DDL in *Parameters and Functions Reference for Oracle GoldenGate*.

Using DDL Statements and Options for Filtering

- INCLUDE (default) means include all objects that fit the rest of the description. EXCLUDE means to omit items that fit the description. Exclude rules take precedence over include rules.
- OPTYPE specifies the types of operations to be included or excluded. You can use CREATE and ALTER. Multiple OPTYPE can be specified using parentheses. For example, OPTYPE (CREATE, ALTER). The asterisk (*) wildcard can be specified to indicate all operation types, and this is the default.
- OBJTYPE specifies the TABLE operations to include or exclude. The wildcard can be specified to indicate all object types, and this is the default.
- OBJNAME specifies the actual object names to include or exclude. For example, eric.*. Wildcards are specified as in other cases where multiple tables are specified. The default is *.
- STRING indicates that the rule is true if any of the strings in STRINGSPEC are present (or false if EXCLUDESTRING is specified and the STRINGSPEC is present). If multiple STRING

entries are made, at least one entry in each STRINGSPEC must be present to make the rule evaluate true.

For example:

```
DDL OPS STRING ("a", "b"), STRING ("c") EVALUATES true IF STRING "a" OR "b"  
IS PRESENT, AND STRING "c" IS PRESENT
```

- LOCAL is specified if you want the rule to apply only to the current Extract trail (the Extract trail to which the rule applies must precede this DDL OPS specification).
- The semicolon is required to terminate the parameter entry.

In the following example, DDL is used to capture CREATE and ALTER for all objects in eric except for those starting with name tab.

```
DDL &  
INCLUDE OPTYPE CREATE &  
INCLUDE OPTYPE ALTER &  
INCLUDE OBJNAME (eric.*) &  
EXCLUDE OBJNAME (eric.tab*);
```

In the following example, the DDL captures CREATE for all objects in joe, alter for all index type objects

```
DDL &  
INCLUDE OPTYPE CREATE &  
INCLUDE OBJNAME (joe.*) &  
INCLUDE OPTYPE ALTER &  
INCLUDE OBJTYPE (index);
```

In the following example, the DDL captures CREATE for all objects in joe.

```
DDL &  
INCLUDE OPTYPE CREATE &  
INCLUDE OBJNAME (joe.*)
```

Note

There can be only one DDL filtering statement. If DDL filter is long, use ampersand (&) sign to continue it on another line.

The DDL OPS.C module stores all DDL operation parameters and executes related rules.

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

See DDL and DDLOPTIONS.

Filtering DDL Generated by MySQL Heatwave Engine

MySQL database service is HeatWave enabled from MySQL database service version 8.0.33 and has RAPID cluster plug in installed. The RAPID cluster is attached to MySQL database service instance when you enable HeatWave on that instance. You can load any base table to HeatWave engine using the following DDL statement:

```
ALTER TABLE mysrc.emp secondary_engine = RAPID;  
ALTER TABLE t1 secondary load;
```

After the table is loaded to the engine, this DDL statement is logged to the MySQL binary log.

- If DDL capturing for this table (mysrc.emp) is enabled on Oracle GoldenGate for MySQL Extract, then the preceding DDL statement is processed by the Extract and the DDL is written to the trail file by the Extract.
- If both source and target are MySQL Heatwave, then the DDL is successfully applied to the target MySQL.
- If the source is MySQL HeatWave and target is non-HeatWave, then Replicat throws an error similar to the following:

```
ALTER TABLE vinsrc.emp1 secondary_load;
```

```
ERROR 1286 (42000): Unknown storage engine 'rapid'
```

To avoid this error, use the following DDL option in the Extract parameter file to filter these queries.

```
DDL INCLUDE MAPPED , EXCLUDE INSTR 'secondary_load';
```

PostgreSQL: DDL Replication

Learn about DDL replication in PostgreSQL.

PostgreSQL: Prerequisites for Configuring DDL

The DDL changes in the source database are not automatically handled by the Extract process in PostgreSQL. Perform the following steps before making any DDL changes on the source table.

1. Stop the application data to the tables for which the DDL needs to be modified.
2. OGG Extract has captured all the changes prior to the DDL change.
3. Stop the Extract.
4. Execute DELETE TRANDATA on the table.
5. Make the required DDL Change using ALTER TABLE ADD COLUMN/DROP COLUMN command.
6. Execute ADD TRANDATA on the table.
7. Start the Extract.
8. Resume the application data on the table.

PostgreSQL: During DDL Replication

The Extract behaves as follows when the extract is in running or in stopped state during DDL Replication:

- If the columns are added when Extract is in stopped state, Extract processes newer change records (after the DDL change). It reports a one-time warning while processing old change records (prior to DDL change) and ignores the missing columns
- If the columns are added when Extract is in running state, Extract processes older change records (prior to DDL change) but it abends when it encounters new change records (after the DDL change), with the newly added columns.
- If the columns are removed when Extract is in stopped state, Extract processes only the columns that are received in the older change records by returning a one-time warning and ignores the extra columns.
- If the columns are removed when Extract is in running state, Extract processes only the columns that are received in the change records by returning a one-time warning for the missing columns.
- Extract processes the columns for which it has the metadata and ignores any extra columns received from the transaction log.

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.

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.

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
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 `DDLSUBST` parameter statement, it might help to review [How DDL is Evaluated for Processing](#).

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)

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.

About Oracle GoldenGate AI Service

The AI Service in Oracle GoldenGate works with Replicat. The Replicat process interacts with an external AI service defined in the ServiceManager. From the ServiceManager, AI service

provider details are configured depending on the type of AI service being embedded. Based on the AI Service provide, AI models are created from the Service Manager.

Note

To ensure security, Oracle GoldenGate requires the use of CA certificates when connecting to any AI service provider. For AI services outside the corporate network, an HTTP proxy must be used.

Configure and Use an AI Service

Oracle GoldenGate provides the AI Service to allow embedding from various AI Service providers, including Google Gemini, OpenAI, Voyager AI, and OCI Generative AI. This service allows you to bring embedded data from the AI service.

Before you begin using the AI Service, make sure that you have the required values to configure the AI Service Provider and the associated AI Model details.

From the Service Manager, first you configure an AI Service Providers and then configure AI Service models based on your AI Service Provider from the Service Manager.

Here are the steps to add an AI Service Provider:

1. Click the plus sign in the Providers section. This displays the Add Provider dialog box to add an AI provider.
2. Specify the details depending on your AI provider. The following AI Providers are supported:

Provider Type	Options
Gemini	Specify the following values: • Provider ID, such as geminiai • Provider Name • Base URL • API Key
OCI Generative AI	Specify the following values: • Provider ID, example oracleai • Provider Name, example, oci • Base URL, example <code>https://generativeai.regionapi_version</code> • Tenancy OCID • Compartment OCID • User OCID • API Key, generate the API Key and upload it in OCI Console. Add this key in the • Key Fingerprint, fingerprint of the API Key.
OpenAI	Specify the following values: • Provider ID, such as geminiai • Provider Name • Base URL • API Key

Provider Type	Options
Voyage AI	Specify the following values: • Provider ID, such as geminiai • Provider Name • Base URL • API Key

3. Click Submit.

The AI Provider is listed in the Providers list. Now, you can create an AI Model associated with the provider.

To create an AI Model, add the following values:

- **Model ID:** Depending on the AI service provider, you can specify a model ID. For example, for Gemini, you can select the model ID as gemini-3.0-pro.
- **Model Name:** Specify a name for the model.
- **Remote Model Name:** Specify the name of the model that is hosted on a provider's server and would be accessed using APIs for different tasks.
- **Provider ID:** This is the same provider that you have created.
- **Capabilities (Embedding):** Preselected option set to Embedding.
- **Enabled (toggle switch):** Enable or disable the model.
- **Maximum Input Characters:** Maximum characters allowed in the given Model.

AI Models

From the left-navigation pane, click the **AI Models** option to view the AI models that are available with the configured AI Providers.

Configure DDL Modification for Oracle GoldenGate for Sybase

Use the following steps to modify DDL for Oracle GoldenGate Extract for Sybase:

1. Pause or stop capturing application data for tables where you want to modify the DDL.
2. Ensure Extract processes all the transactions prior to making any DDL changes. An Event Marker table may help to ensure full completion.
3. Stop Extract.
4. At source, execute `DELETE TRANDATA` for the tables on which DDL has to be performed.
5. Execute the `ALTER TABLE` statement on the database side, to add or drop the column in or from the tables.
6. Re-enable trandata using the `ADD TRANDATA` command for the same tables at source.
7. Start Extract.

Using Oracle GoldenGate with MySQL Group Replication

This topic describes the requirements and configuration steps for setting up Oracle GoldenGate to support MySQL Group Replication.

Oracle GoldenGate Features to Support MySQL Group Replication

The following are Oracle GoldenGate features required to support capture from a MySQL database Group Replication instance.

CSN Format

The Extract for MySQL Group Replication uses a new CSN format that is based on the Group Replication Global Transaction ID. This CSN format should be used with ATCSN and AFTERCSN when manually positioning a MySQL Group Replication Extract or Replicat whose source trail was generated by a MySQL Group Replication Extract.

An example of the sequence used in group replication capture is:

```
00000000000000000001:f77024f9-f4e3-11eb-a052-0021f6e03f10:0000000000000010654
```

In this sequence, the Oracle GoldenGate sequence number is 00000000000000000001 and the GTID is f77024f9-f4e3-11eb-a052-0021f6e03f10:0000000000000010654.

Extended Checkpoint Support

The Extract for MySQL Group Replication includes an extended checkpoint file in addition to the core Extract checkpoint file. The extended checkpoint file is created in the same checkpoint directory where the core checkpoint and has a cpex extension after the name of the capture group for example, extmysql.cpex.

This file is created when Extract starts and is deleted when Extract is deleted and should not be edited.

Using GTID-based Extract

If `gtid_mode` is enabled in MySQL database, then Oracle GoldenGate Extract for MySQL automatically starts using the GTID-based recovery mechanism and extended checkpoint, which enables it to support failover and recovery. There is no extra parameter required for the Extract.

Note

If not using Group Replication, it is recommended to disable `gtid_mode` on the source MySQL database. This will return the Extract's capture behavior to using the log number and offset method.

Position by GTID set in Oracle GoldenGate GTID-based Extract for MySQL

A new position by GTID set option is added in the GTID-based Extract for MySQL. Positioning by the GTID set is supported only for GTID-based capture in MySQL.

An introduction to the GTIDs and the GTID set can be found in MySQL database documentation:

<https://dev.mysql.com/doc/refman/8.0/en/replication-gtids-concepts.html>

The MySQL supported sources for this feature are MySQL Server 8.0, MySQL Server 5.7, and MySQL Database Service (MDS).

The maximum supported GTID set size is 64 KB.

See `ADD EXTRACT` and `ALTER EXTTRAIL` for details on the `GTIDSET` parameter usage.

Also see [REST API Example for GTIDSET parameter](#).

REST API Example for GTIDSET parameter

This topic shows examples for using the GTIDSET parameter with ADD EXTRACT REST API post request, ALTER EXTRACT REST request, and then checking the Extract with the GTID set value using the INFO EXTRACT command.

Example 1: The REST API post request to add an Extract is as follows:

```
{
  "credentials": {
    "alias": "ggeist"
  },
  "source": "tranlogs",
  "begin": {
    "at": {
      "atGtidSet": [
        "2174B383-5441-11E8-B90A-C80AA9429562:1-3",
        "3174B383-5441-11E8-B90A-C80AA9429562:1-3",
        "4174B383-5441-11E8-B90A-C80AA9429562:1-3"
      ]
    }
  }
}
```

The example REST request to add Extract is formed:

```
> POST /services/{version}/extracts/exte
{
  "source": "tranlogs",
  "begin": {
    "at": {
      "atGtidSet": "1d380bee-0c24-11ee-b338-00001701f1ea:1-8"
    }
  }
}
```

The example REST request to alter Extract is formed as follows:

```
> PATCH /services/{version}/extracts/exte
{
  "begin": {
    "at": {
      "atGtidSet": "1d380bee-0c24-11ee-b338-000017015555:12-19"
    }
  }
}
```

After adding the Extract by GTID set, use the INFO EXTRACT command with showch to view the GTID set as position, as shown in the following example:

```
OGG (http://localhost:9011/ GTIDMAIN) 6> info extract exte, showch
```

```
Extract      EXTE      Initialized  2023-07-17 19:00      Status STOPPED
```



```
Checkpoint Lag      00:00:00 (updated 00:00:06 ago)
VAM Read Checkpoint First Record
00000000000000000000:cd6f98b1-1fbd-11ee-b73f-00001701f1ea:20
GTID Set            cd6f98b1-1fbd-11ee-b73f-00001701f1ea:1-20
Encryption Profile  LocalWallet
```

Requirements for Supporting Group Replication

This topic describes the requirements for using Oracle GoldenGate with MySQL Group Replication database clusters.

- Oracle GoldenGate for MySQL Group Replication supports MySQL version 8.0 or higher.
- Only Group Replication configured in Single-Primary Mode is supported for Extract.
- Only the `gtid_mode = ON` is supported in the GTID-based capture.

Limitations of Group Replication with Oracle GoldenGate for MySQL

Here are the limitations of support when running group replication with Oracle GoldenGate for MySQL:

- Oracle GoldenGate GTID-based capture does not support tags.
- Oracle GoldenGate Extract with MySQL Group Replication does *not* support multi-primary Group Replication mode.
- Oracle GoldenGate Extract with MySQL Group Replication does not support writing to remote trails. If using RMTTRAIL with an Extract, the Extract will abend with the following error:

```
"Trail file ea is remote. Only local trail allowed for this extract."
```

In this example, ea is the remote trail file name.

Use the DISTPATH process to transport remote trails. See [Monitor the Path Information and Statistics](#).

- Oracle GoldenGate Extract with MySQL Group Replication does not support multiple trails.

SSL Configuration on Group Replication Cluster

Learn about SSL configuration on Group Replication Cluster.

Overview of Database Cluster SSL Configuration for Group Replication

A clustered database environment contains different nodes, constituting one primary node and one or more secondary nodes. There can be only one primary node at any instant. Each node has its own distinct hostname with a MySQL database instance, which is maintained by a separate configuration for that particular node. All the nodes in the cluster collectively represent the database.

There is a Router as well, which is the first point of contact for any client trying to connect to the database.

When enabling SSL connectivity, all of the database nodes and the Router will need to have their own authorization keys and server certificates. These certificates must be authorized by a common Certificate Authority (CA).

The certificates that are commonly used for this setup are:

- `ca.pem`: The certificate of the common CA (Certification Authority)
- `server-cert.pem`: The certificate that is certified by the CA for identifying the database node
- `server-key.pem`: The private key of the individual database node
- `router-cert.pem`: The certificate that is certified by the CA for identifying the router
- `router-key.pem`: The private key of the router

Configuration for the Router and database nodes is described in the following tables. For the purpose of this explanation, the following example shows one router and three database nodes.

Table 9-1 Router and Database Node Configuration

Router	-
Hostname	<code>mysqlrouter.company.com</code>
Config Filename	<code>mysqlrouter.conf</code>
Port	6446
Common Name	<code>mysqlrouter.company.com</code>
Certificate Name	<code>server-cert.pem</code>
Key file name	<code>server-key.pem</code>
Database Node 1	-
Hostname	<code>dbnode1.company.com</code>
Config Filename	<code>my.cnf</code>
Port	3308
Common Name	<code>dbnode1</code>
Certificate Name	<code>server-cert.pem</code>
Key file name	<code>server-key.pem</code>
Node Rank	Primary
Database Node2	-
Hostname	<code>dbnode2.company.com</code>
Config Filename	<code>my.cnf</code>
Port	3308
Common Name	<code>dbnode2</code>
Certificate Name	<code>server-cert.pem</code>
Key file name	<code>server-key.pem</code>
Node Rank	Secondary
Database Node3	-
Hostname	<code>dbnode3.company.com</code>
Config Filename	<code>my.cnf</code>
Port	3308
Common Name	<code>dbnode3</code>

Table 9-1 (Cont.) Router and Database Node Configuration

Router	-
Certificate Name	server-cert.pem
Key file name	server-key.pem
Node Rank	Secondary

Create Server Certificates

Before you begin configuring the router and database nodes, you'll need to create SSL server certificates. For connecting database nodes and router using SSL, you must have the right SSL keys and certificates for secure communication. All certificates must be recognized by a common Certification Authority (CA). If the keys and certificates were auto-generated during database/router installation (or if they are self-signed) then the connection might fail. Only certificates that are authorized by a CA are allowed to proceed further.

If the authorized server key and certificates are already available, then ensure that the certificates have the correct permissions and have been placed in the correct path for the router/database node.

For steps to generate SSL certificates for server, see:

[Creating SSL Certificates and Keys Using OpenSSL](#)

Tasks for Configuring SSL Certificates

1. Generate a separate certificate and key for each database node.
2. Use the same ca.pem which is common to all database nodes and routers.
3. In the server-certificate for the database nodes, specify the common name *without* the domain name. See the common name in the [Table 9-1](#) in [Overview of Database Cluster SSL Configuration for Group Replication](#) for reference.
4. Ensure that the server **certificate name** and **key file name** match the corresponding database node and router values.
5. To verify the CN values in each generated server certificate, invoke openssl using the following commands :

```
openssl x509 -text -in ca.pem
openssl x509 -text -in server-cert.pem
openssl x509 -text -in client-cert.pem
```

The issuer CN must be the same for all. The subject CN must contain only hostname without domain name.

6. After generating the certificates, verify them against the CA file.
7. Copy the generated certificate and key file to the MySQL data directory for each database node and router. Ensure that you provide read permission to all users and retain write permission to file owner only.
8. Copy the common ca.pem to every node and router and provide read permissions to all users.

Configure Database Nodes and Router

Use the settings similar to the following, to configure database nodes and router for connecting over a secure SSL connection.

Router

In the Router config file, ensure that the below settings are present:

```
CLIENT_SSL_MODE=PREFERRED
CLIENT_SSL_CERT=absolute path of the generated router certificate
CLIENT_SSL_KEY=absolute path of the generated router key
SERVER_SSL_MODE=AS_CLIENT
SERVER_SSL_VERIFY=VERIFY_IDENTITY
SERVER_SSL_CA=absolute path of the common ca.pem placed on this server
```

After it is configured, provide read permissions to all users and revoke write permissions from group and others.

Database Node

In each of the MySQL database nodes, make sure the following are set under the appropriate section:

```
SSL_CAPATH=absolute path of the common ca.pem placed on this node
SSL_CA=ca.pem
SSL_CERT=server-cert.pem
SSL_KEY=server-key.pem
GROUP_REPLICATION_SSL_MODE=REQUIRED
REQUIRE_SECURE_TRANSPORT=ON
```

After configuring the database node, provide read permissions to all users and revoke write permissions from group and others.

Testing the Connection

After the configurations are in place and the appropriate permissions have been provided to the configuration files, test the settings by restarting the database nodes and router.

Test the Database Nodes Connection

Ensure that the database node does not terminate. Check the logs under log-error setting in the configuration file for any errors or warnings that indicate the SSL settings were not accepted. Try connecting to the specific node using the following command line (use the common name as specified in the certificate for this node):

```
mysql -u username -p password -h db_common_name -P db_port --ssl-  
mode=VERIFY_IDENTITY --ssl-ca=path/of/ca.pem
```

Make sure that the connection does not generate any errors.
Similarly, connect with different SSL-modes by providing the appropriate parameter values.

Note

The `ssl-cert` and `ssl-key` are not mandatory for `VERIFY_IDENTITY`. However, if the database user requires X509 authentication, then both `ssl-cert` and `ssl-key` must be provided with `client-cert` and `client-key`.

Test all database nodes using this method and then test the router connection.

Test the Router Connection

After the database nodes are up, restart the router and monitor it ensuring it does not terminate.

Check the logs under `log-error` setting in the configuration file for any errors or warnings that indicate the SSL settings were not accepted. If there are no errors or warnings, try connecting to the database from the router using the following command. Make sure you use the common name as specified in the certificate for the router:

```
mysql -u username -p password -h router_common_name -P router_port --ssl-  
mode=VERIFY_IDENTITY --ssl-ca=path/of/ca.pem
```

Ensure that connection goes through without any errors.

Verify the Connection from the Router to the Database Node

First determine the currently active primary node, using the following command:

```
MySQL> SHOW VARIABLES like '%hosts%';
```

Now logout from the database and switchover the database to another node. Then login to the database from the router again, using the following command:

```
mysql -u username -p password -h router_common_name -P router_port --ssl-  
mode=VERIFY_IDENTITY --ssl-ca=path/of/ca.pem
```

Check the currently active primary node using the same command again:

```
MySQL> SHOW VARIABLES like '%hosts%';
```

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.');
```

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.

View	Description
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
SQLEXEC 'call <i>procedure_name</i> ()'	Execute a stored procedure
SQLEXEC 'sql_query'	Execute a query
SQLEXEC 'database_command'	Execute a database command

Argument	Description
'call <i>procedure_name</i> ()'	Specifies the name of a stored procedure to execute. The statement must be enclosed within single quotes. Example: SQLEXEC 'call prc_job_count ()'
'sql_query'	Specifies the name of a query to execute. The query must be contained all on one line and enclosed within single quotes. 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. SQLEXEC 'SELECT "col1" from "schema"."table"'
'database_command'	Specifies a database command to execute. Must be a valid command for the database.

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 TABLE or MAP 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 SELECT statement or change the database with an INSERT, UPDATE, or DELETE 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"'	

Argument	Description
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 9-2 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 REPERROR parameter specified in the Replicat parameter file. Oracle GoldenGate continues processing other stored procedures or queries associated with the current TABLE or MAP statement before processing the error.
FINAL	Performs in a similar way to RAISE 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.

Mapping and Manipulating Data

Learn about tasks, functions, commands, and processes used for integrating data between source and target tables.

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 9-3 Sample Column Mapping

Parameter statement	Description
COLMAP	Begins the COLMAP statement.
USEDEFAULTS,	Maps source columns as-is when the target column names are identical.
NAME = CUST_NAME,	Maps the source column CUST_NAME to the target column NAME.
TRANSACTION_DATE = @DATE ('YYYY-MM-DD', 'YY', YEAR, 'MM', MONTH, 'DD', DAY),	Converts the transaction date from the source date columns to the target column TRANSACTION_DATE by using the @DATE column conversion function.
AREA_CODE = @STREXT (PHONE_NO, 1, 3), PHONE_PREFIX = @STREXT (PHONE_NO, 4, 6), PHONE_NUMBER = @STREXT (PHONE_NO, 7, 10)) ;	Converts the source column PHONE_NO into the separate target columns of AREA_CODE, PHONE_PREFIX, and PHONE_NUMBER by using the @STREXT 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 ("Co1A" = Co1A)
```

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 columns, 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>	Maps based on column names. 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. For databases that support case-sensitivity without requiring quotes, specify the column name as it is stored in the database. If the COLMATCH 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. COLMATCH NAMES "aBc" = aBc
<code>PREFIX prefix SUFFIX suffix</code>	Ignores the specified name prefix or suffix. 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. 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. 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. COLMATCH PREFIX 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". SUFFIX "_k"
<code>RESET</code>	Turns off previously defined COLMATCH rules for subsequent TABLE or MAP statements.

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 VARCHAR	Spaces
Date or Datetime	Current date and time
Columns that can take a NULL 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 9-4 Using Multiple FILTER Statements

Parameter file	Description
REPERROR (9999, EXCEPTION)	Raises an exception for the specified error.
MAP OWNER.SRCTAB, TARGET OWNER.TARGETAB,	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 9-4](#), 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 9-5](#) 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 9-5 Permissible WHERE Operators

Element	Examples
Column names	PRODUCT_AMT
Numeric values	-123, 5500.123

Table 9-5 (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 9-6 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 9-6](#) 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 9-7 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 9-8 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 9-9 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 9-9 (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.ora_test, 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.

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.

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 CREATES 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.listFiles) 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;
```

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.

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 9-21 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 9-22 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.

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 ggseerr.log file within the root Oracle GoldenGate directory
- VIEW GGSEVT command.

You can control the ggseerr.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-ggseerr.xml file in one of the following locations:

Microservices Architecture
\$OGG_HOME/etc/conf/logging/

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

Following are the key requirements for configuring Automatic CDR in Oracle GoldenGate:

- 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 are 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 the unique key (UK) with at least one non-nullable field, if there is no PK.

- SITE PRIORITY needs support from the Replicat, before the parameters are implemented and the setting is passed to the Replicat.
- Automatic CDR does not work with other MAP level exceptions and error handling configurations, such as REPERROR, MAPEXCEPTION. You can either use Automatic CDR or

REPERROR/MAPEXCEPTION but not both on the same table. See TABLE | MAP and REPERROR to know more.

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.

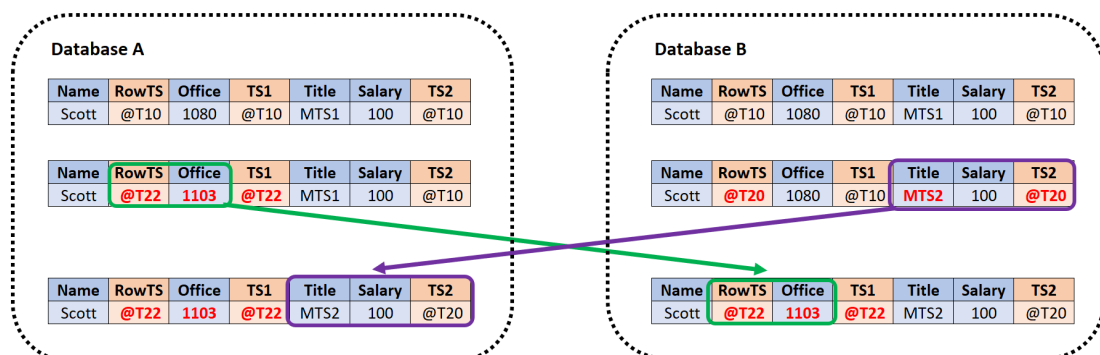
Column Groups

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.

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

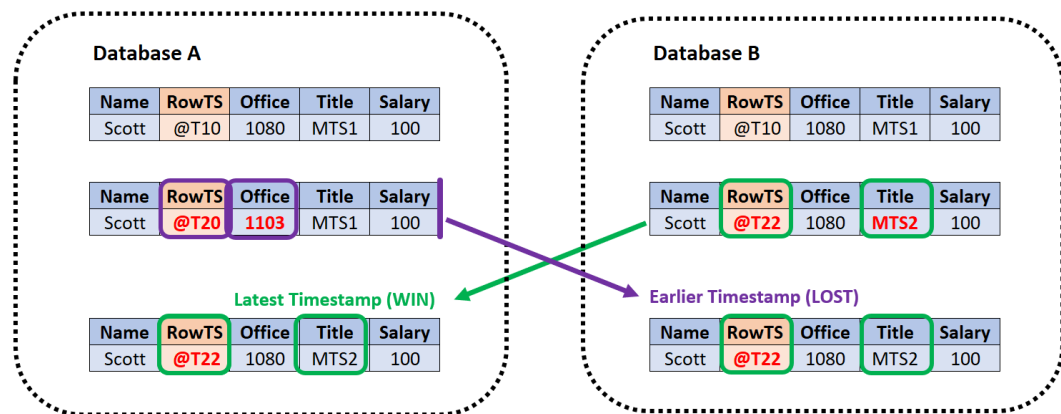
When you run the ADD_AUTO_CDR procedure in the DBMS_GOLDENGATE_ADM package to configure a table for automatic Oracle GoldenGate conflict detection and resolution, a hidden timestamp column is added to the table. This hidden timestamp column records the time of a row change, and this information is used 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.	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. 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.

Operation	Conflict Detection	Conflict Resolution
UPDATE	A conflict is detected in each of the following cases: - 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.	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. 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. 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. 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. 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.
DELETE	A conflict is detected in each of the following cases: - 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.	If the timestamp of the row LCR is later than the timestamp in the table, then delete the row from the table. 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. If the delete is successful, then log the row LCR by inserting it into the tombstone table. If the table row does not exist, then log the row LCR by inserting it into the tombstone table.

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.

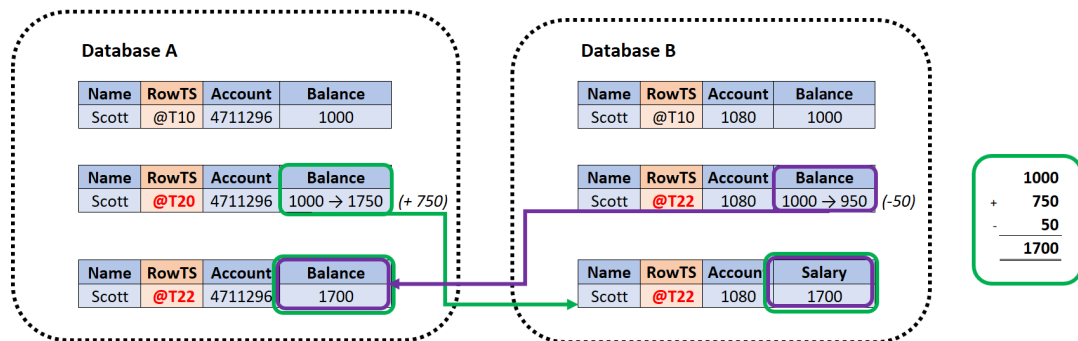
Delta Conflict Resolution

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.

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.

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.

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 9-23 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 AI Database 26ai 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 9-24 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 9-25 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 9-26 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 9-27 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 9-28 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 9-29 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');
```

Online Redefinition on ACDR Tables

Oracle AI Database allows mitigating application interaction when reorganizing tables or columns using the DBMS_REDEFINITION package. The DBMS_REDEFINITION package can be used to perform the following tasks:

- Remove unused columns.
- Reorganization, tablespace redesign, and partitioning.

The DBMS_REDEFINITION.START_REDEF_TABLE automatically manages the hidden timestamp column to the interim table.

For details about the DBMS_REDEFINITION Package, see the DBMS_REDEFINITION in the *Oracle Database PL/SQL Packages and Types Reference*.

Removing Conflict Detection and Resolution From a Table

With Oracle AI Database 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 instance that replicates the table.

Example 9-30 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 are 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 AI Database 26ai 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 9-31 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 9-32 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 9-33 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 9-34 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 9-35 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 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.TABLENAME', 'CDR_CONFLICTS')
@GETENV ('STATS', 'TABLE', 'SCHEMA.TABLENAME', 'CDR_RESOLUTIONS_SUCCEEDED')
@GETENV ('STATS', 'TABLE', 'SCHEMA.TABLENAME', '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 DELETEROWMISSING 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 9-10 INSERTROWEXISTS Conflict with USEMAX Resolution

Image	SQL	Comments
Before image in trail	None (row was inserted on the source).	N/A

Table 9-10 (Cont.) INSERTROWEXISTS Conflict with USEMAX Resolution

Image	SQL	Comments
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>	last_mod_time='9/1/10 3:00' 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	<pre> name='Mary' phone='111111' address='Ralston' salary=200 balance=500 comment='aaa' last_mod_time='9/1/10 1:00' </pre>	last_mod_time='9/1/10 1:00' is the current image of the resolution column in the target against which the resolution column value in the trail is compared.
Initial INSERT applied by Replicat that detects the conflict	SQL bind variables: <pre> 1)'Mary' 2)'1234567890' 3)'Oracle Pkwy' 4)100 5)100 6)NULL 7)'9/1/10 3:00' </pre>	This SQL returns a uniqueness conflict on 'Mary'.
UPDATE applied by Replicat to resolve the conflict	SQL bind variables: <pre> 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' </pre>	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 9-11 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.
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 9-12 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)	
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 9-13 DELETEROWEXISTS Conflict with OVERWRITE Resolution

Image	SQL	Comments
Before image in trail	<pre> name='Mary' phone='222222' address='Holly' salary=100 balance=100 comment=NULL last_mod_time='9/1/10 5:00' </pre>	N/A
After image in trail	None	N/A
Target database image	<pre> name='Mary' phone='1234567890' address='Oracle Pkwy' salary=100 balance=600 comment=com last_mod_time='9/1/10 7:00' </pre>	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.
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).

DELETEROWMISSING 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 9-14 DELETEROWMSING Conflict with DISCARD Resolution

Image	SQL	Comments
Before image in trail	<pre> name='Jane' phone='4444' address='Holly' salary=200 balance=200 comment=NULL last_mod_time='9/1/10 8:00' </pre>	N/A

Table 9-14 (Cont.) DELETEROWMSING Conflict with DISCARD Resolution

Image	SQL	Comments
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 DELETEROWMISSING, 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 9-15 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>	last_mod_time='9/1/10 3:00' is the before image of the resolution column for the USEMAX resolution. salary=100 is the before image for the USEDELTA resolution.
After image in trail	<pre>phone='222222' address='Holly' salary=200 comment='new' last_mod_time='9/1/10 5:00'</pre>	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.

Table 9-15 (Cont.) UPDATEROWEXISTS with USEDELTA and USEMAX

Image	SQL	Comments
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>	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. salary=600 is the current image of the target column for the USEDELTA resolution.
Initial UPDATE applied by Replicat that detects the conflict	SQL bind variables: <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)'9/1/10 3:00' </pre>	This SQL returns a no-data-found error because the values for the salary and last_mod_time are different. (The values for comment and balance are also different, but these columns are not compared.)
UPDATE applied by Replicat to resolve the conflict for salary, using USEDELTA.	SQL bind variables: <pre> 1)200 2)100 3)'Mary' </pre>	Per USEDELTA, the difference between the after image of salary (200) in the trail and the before image of salary (100) in the trail is added to the current value of salary 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 USEMAX.	SQL bind variables: <pre> 1)'222222' 2)'Holly' 3)'new' 4)'9/1/10 5:00' 5)'Mary' 6)'9/1/10 5:00' </pre>	Per USEMAX, because the after value of last_mod_time 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 salary column is not set here, because it is resolved with the UPDATE from the USEDELTA 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 9-16 UPDATEROWEXISTS with USEDELTA, USEMAX, and IGNORE

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 for the USEMAX resolution. salary=100 and balance=100 are the before images for the USEDELTA resolution.
After image in trail	phone='222222' address='Holly' salary=200 comment='new' last_mod_time='9/1/10 5:00'	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. salary=200 is the only after image available for the USEDELTA resolution. For balance, the before image will be used in the calculation.
Target database image	name='Mary' phone='1234567890' address='Ralston' salary=600 balance=600 comment='com' last_mod_time='9/1/10 4:00'	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. salary=600 and balance=600 are the current images of the target columns for USEDELTA.

Table 9-16 (Cont.) UPDATEROWEXISTS with USEDELTA, USEMAX, and IGNORE

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) 100 11) '9/1/10 3:00'	This SQL returns a no-data-found error because the values for the address, salary, balance and last_mod_time columns are different.
UPDATE applied by Replicat to resolve the conflict for salary, using USEDELTA.	SQL bind variables: 1) 200 2) 100 3) 'Mary'	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.
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 last_mod_time in the trail record is greater than the current value in the database, that column plus the address column are updated with the after values from the trail record. Note that the salary 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 (phone and comment), so no resolution SQL is applied.

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: HANDLECOLLISIONS, 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 HANDLECOLLISIONS 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 REPERROR.
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.

```

-- REPERROR error handling: DEFAULT represents all error types. DISCARD
-- writes operations that could not be processed to a discard file.
REPERROR (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 REPERROR.
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.

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

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.

$$[\text{source transaction log volume in one hour}] \times [\text{number of hours downtime}] \times .4$$
$$= \text{trail disk space}$$

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](#).

Change the Size of Trail Files

You can use the Oracle GoldenGate web interface, Admin Client, or REST API to change the trail file size.

To change the trail file size from the Admin Client, you can use the MEGABYTES option of either the ALTER EXTTRAIL command. To change the file size, follow this procedure.

1. Issue one of the following commands, depending on the location of the trail, to view the path name of the trail you want to alter and the name of the associated Extract group. Use a wildcard to view all trails.

```
INFO EXTTRAIL *
```

2. Issue one of the following commands, depending on the location of the trail, to change the file size.

```
ALTER EXTTRAIL trail, EXTRACT group, MEGABYTES n
```

3. Issue the following command to cause Extract to switch to the next file in the trail.

```
SEND EXTRACT group, ROLLOVER
```

Download Trail Files

Starting with Oracle GoldenGate 23ai, you can download trail files to your local system. This might be required if you need to do additional analysis or you need more information for diagnostic purposes.

Within your local Oracle GoldenGate environment, you can use logdump to access the trail file records. See [Using the LogDump Utility to Access Trail File Records](#).

To begin downloading trail files, you need to perform the following steps:

1. Log in to the Administration Service.
2. From the left navigation pane, click Trails to open the Trail page in the right pane. This page displays the Trails table that provides various details about the trail files, including the Extract (producer) and Replicat (consumer), last archived sequence number among others. You can monitor the trail file from this table.
3. To download the trail file, click the Download icon from the Action column for the trail file that you want to download. The Download Trail Sequence dialog box is displayed. From this dialog box, you can specify a trail sequence number from where the trail file will be downloaded.
4. After determining the required trail sequence number, enter the value in the Sequence Number box and click **Submit**.

The trail files are downloaded to the specified location at the host system from where the call is initiated.

Note

You can download only one trail file at a time.

You can also use the cURL command to download the trail file:

```
curl -s -k -u ggma:**** --basic -o ea000000096\  
-H 'Accept: application/binary' \  
-X GET "https://north.oraclevcn.com:9200/services/v2/trails/ea/  
sequences/96?path=north"
```

Delete Trail Files

Starting with Oracle GoldenGate 23ai, trail files can be deleted on your local system. Deleting trail sequences can be done in two ways from the Trails page of the Administration Service:

- Delete all trail files.
- Specify a range of trail sequences to be deleted.

Both options are available and can be used as follows:

1. Log in to Administration Service.
2. From the left navigation pane, click Trails to open the Trails page.
3. From the Action column of the Trails table, click the Delete icon to open the Delete Trails Sequences dialog box.
4. Select the **All** option to delete all the trail files generated to the current timestamp.
5. If you want to delete only a section of the trail file, then select the **Range of Sequences** option and specify the starting sequence number and end sequence number for the range of the trail file to be deleted.
6. Click **Delete** to complete the deletion process.

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.

Archive Trails

You can choose to archive trails at a specified location, instead of purging trail files. This automation task allows you to automatically archive trail files to a specified location after a certain time duration. To automate archival of trail files:

1. From the left navigation pane, click **Tasks** and select **Archive Trails**.
2. In the Archive Trails page, click the plus (+) sign next to the Archive Trails Task to open the **Create a new Archive Trail Task** dialog box.
3. Enter the following details in the **Create a new Archive Trail Task** dialog box:
 - Name: Specify a name for the task.
 - Enabled: This toggle switch is enabled by default. You can choose to disable it if you want to keep the task but not activate it.
 - Trails: Specify the name of the trail file for archiving.
 - Archive Subset of Sequence: If you want to archive only a certain sequence from the trail file, then enable this toggle switch. When you enable this option, the Sequence Start and Sequence End boxes get enabled. Specify the sequence start and end values in these text boxes.
 - Archive Target Type: The target type stays as File System but you can specify a subdirectory where the archived trail files are stored.
 - Archive Frequency: Specify the frequency interval for archiving the trail file. The trail file is archived automatically when the specified interval completes.
4. Click **Submit** to create the task.
5. From the **Action** column of the Archive Trail Tasks table, you can edit or delete the task using the icons.

Purge Trails

When you automate the purge trail task, Oracle GoldenGate periodically removes trail files that were processed or consumed. Automating this task ensures that the trail files are deleted after the specified interval to avoid excessive consumption of disk space. To set up automate purging of trails:

1. From the Administration Service left navigation pane, click **Tasks** and select **Purge Trails**.
2. Add a Purge Trail task by clicking the + sign .
3. Assign a name to the task in the **Name**. The operation name is case sensitive. For example, you can create an operation with the name **TASK1** and another operation named **task1**.
4. Enter the trail path or trail name in the **Trail** field.
5. Click the + sign to add the trail to the **Selected Trails** list.
6. 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.
7. 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.
8. Specify the number of hours or days when the purge trails task has to run, in the Purge Frequency field and click Submit.

9. Use the Purge Trails task table to edit or delete the task, as required.

Also see PURGE EXTTRAIL.

Purge Process

You can automatically purge processes such as ER, Extract, or Replicat after a specific duration using this automation task. To configure automated purging of Oracle GoldenGate processes:

1. From the Administration Service left navigation pane, click **Tasks** and select **Purge Process**.
2. Enter a name for the task in the **Name** box.
3. 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.
4. Select the Extract or Replicat task (initial load) **Process Type** for the operation.
5. If you enable **Use Stop Status**, the status of the task is used to perform the purge task.
6. Enter the hours or days after which you need to purge the process and click **Submit**.
7. Edit or delete the purge process task using the relevant icon from the Purge Tasks table.

Using Automatic Trail File Recovery

The trail recovery process has the ability to, in some cases, automatically rebuild trail files that are corrupt or missing by Oracle GoldenGate.

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.

About Oracle GoldenGate AI Service

The AI Service in Oracle GoldenGate works with Replicat. The Replicat process interacts with an external AI service defined in the ServiceManager. From the ServiceManager, AI service provider details are configured depending on the type of AI service being embedded. Based on the AI Service provide, AI models are created from the Service Manager.

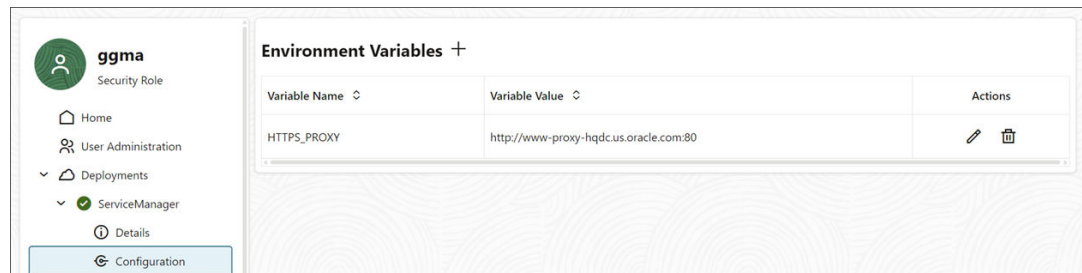
Note

To ensure security, Oracle GoldenGate requires the use of CA certificates when connecting to any AI service provider. For AI services outside the corporate network, an HTTP proxy must be used.

Before Adding an AI Service Provider

Before you add an AI Service Provider from the Service Manager, consider the following:

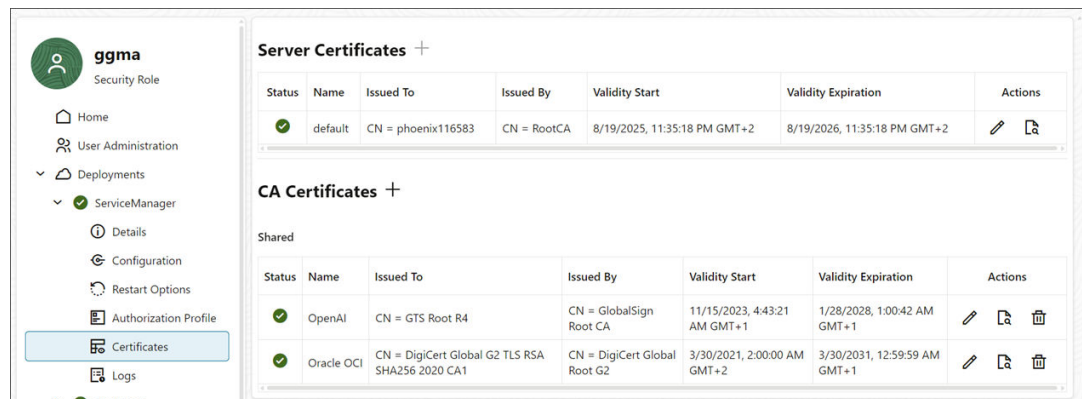
- **Add HTTPS_PROXY environment variable:** If connecting from a corporate network to the public Internet and using OpenAI or other non-Oracle AI Services, then make sure you add the HTTP_PROXY environment variable from the Service Manager. The following diagram shows the HTTP_PROXY environment variable added to the Service Manager configuration. To learn how to add environment variables, see [Access Configuration and Log Details from the Service Manager and Microservices](#).



- **Add certificates for AI Service Providers:** Use the following command to retrieve CA certificates:

```
openssl s_client -showcerts -connect server_name:443
```

You can also add certificates from the Service Manager web interface, as shown in the following image:



Configure and Use the AI Service

Oracle GoldenGate provides the AI Service to allow embedding from various AI Service providers, including Google Gemini, OpenAI, Voyager AI, and OCI Generative AI. This service allows you to bring embedded data from the AI service.

Before you begin using the AI Service from Oracle GoldenGate, make sure to complete the tasks given in [Before Adding an AI Service Provider](#).

You also need to have the required values to configure the AI Service Provider and the associated AI Model details.

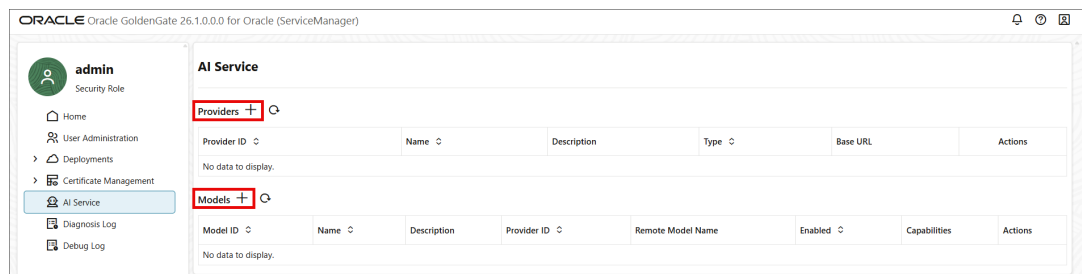
The following table lists the AI Service Provider and the configuration details required for configuring the specific provider:

Provider Type	Options
• Gemini • OpenAI • Voyage AI	The following values are required: • Provider ID • Provider Name • Base URL • API Key
OCI Generative AI	The following values are required: • Provider ID, example oracleai • Provider Name, example, oci • Base URL, example <code>https://generativeai.regionapi_version</code> • Tenancy OCID • Compartment OCID • User OCID • API Key, generate the API Key and upload it in OCI Console. Add this key in the • Key Fingerprint, fingerprint of the API Key.

From the Service Manager, first you configure an AI Service Providers and then configure AI Service Models based on your AI Service Provider.

Here are the steps to add AI Service Providers and AI Models:

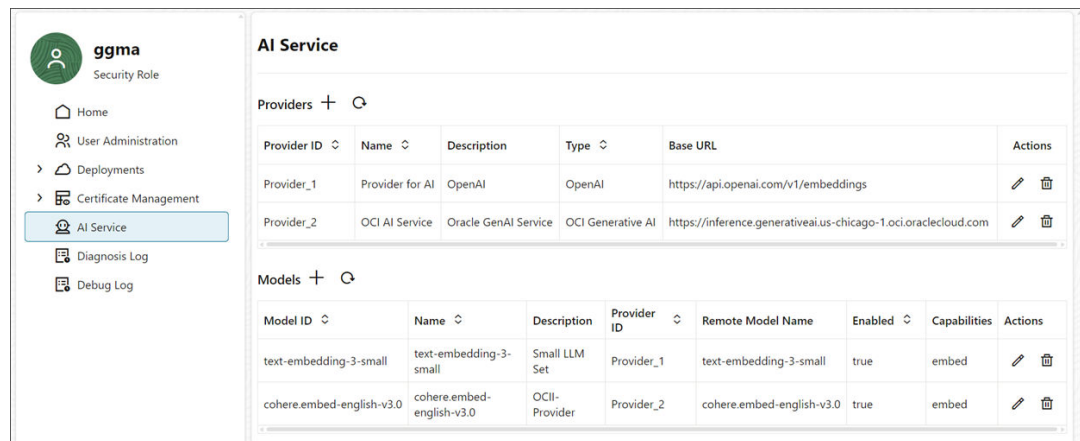
1. Click the plus sign in the **Providers** section. This displays the Add Provider dialog box to add an AI provider.



2. Select from the list of AI Service Providers (Gemini, OpenAI, OCI Generative AI, and Voyager AI) and provide details.
3. Click **Submit**. The AI Provider is listed in the Providers list. Now, you can create an AI Model associated with the provider.
4. To create an AI Model, add the following values:
 - **Model ID**: Depending on the AI service provider, you can specify a model ID. For example, for Gemini, you can select the model ID as gemini-3.0-pro.
 - **Model Name**: Specify a name for the model.
 - **Remote Model Name**: Specify the name of the model that is hosted on a provider's server and would be accessed using APIs for different tasks.
 - **Provider ID**: This is the same provider that you have created.
 - **Capabilities (Embedding)**: Preselected option set to Embedding.
 - **Enabled (toggle switch)**: Enable or disable the model.

- **Maximum Input Characters:** Maximum characters allowed in the given Model.

The following image shows the list of added service Providers and Models on the AI Service page:



The screenshot shows the 'AI Service' page. On the left is a navigation menu with 'AI Service' selected. The main content area has two sections: 'Providers' and 'Models'.

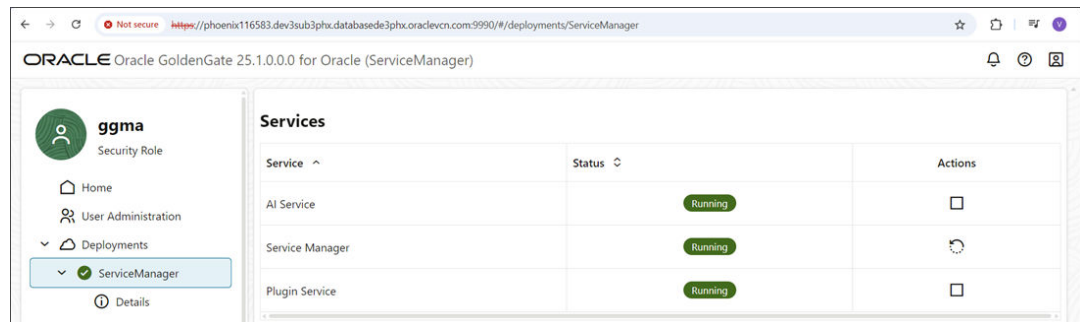
Providers Table:

Provider ID	Name	Description	Type	Base URL	Actions
Provider_1	Provider for AI	OpenAI	OpenAI	https://api.openai.com/v1/embeddings	[Edit] [Delete]
Provider_2	OCI AI Service	Oracle GenAI Service	OCI Generative AI	https://inference.generativeai.us-chicago-1.oci.oraclecloud.com	[Edit] [Delete]

Models Table:

Model ID	Name	Description	Provider ID	Remote Model Name	Enabled	Capabilities	Actions
text-embedding-3-small	text-embedding-3-small	Small LLM Set	Provider_1	text-embedding-3-small	true	embed	[Edit] [Delete]
cohere.embed-english-v3.0	cohere.embed-english-v3.0	OCI-Provider	Provider_2	cohere.embed-english-v3.0	true	embed	[Edit] [Delete]

5. After the AI Service is enabled, it is displayed under the Services list of the Service Manager, as shown in the following image:



The screenshot shows the 'Service Manager' page. The left navigation menu has 'Service Manager' selected. The main content area shows a table of services.

Services Table:

Service	Status	Actions
AI Service	Running	[Stop]
Service Manager	Running	[Refresh]
Plugin Service	Running	[Stop]

6. To view the AI Models, navigate to the Microservices portal and click AI Service from the left navigation pane. The AI models are automatically listed here, as shown in the following image:

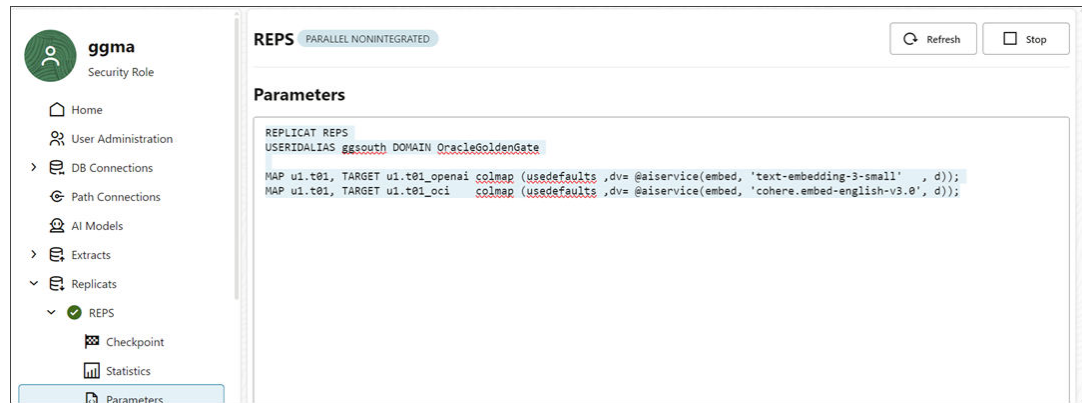


The screenshot shows the 'AI Models' page. The left navigation menu has 'AI Models' selected. The main content area shows a table of AI models.

AI Models Table:

Model ID	Name	Description	Provider ID	Remote Model Name	Capabilities
text-embedding-3-small	text-embedding-3-small	Small LLM Set	Provider_1	text-embedding-3-small	embed
cohere.embed-english-v3.0	cohere.embed-english-v3.0	OCI-Provider	Provider_2	cohere.embed-english-v3.0	embed

7. Move to the **Replicat Parameters** page to see the added parameters, shown in the following image:



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. The name of the Admin Client program is adminclnt.

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. You can create an alias for the username and password and use the alias name in your script to connect to the server.

```
OGG (not connected) 1> ADD CREDENTIALS admin USER ggadmin PASSWORD  
*****
```

Here an alias called admin is created for the actual user ggadmin and password ***** on the server.

Now connect to the service from the Admin Client using the below command. Adminclient gets the real username and password from the wallet in the \$HOME directory of the user executing the script and use those credentials to connect to the service.

```
CONNECT http(s)://my.gg.server.com AS admin
```

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 an alias name for the username and password and use that to connect to the deployment from the Admin Client:

```
ADD CREDENTIALS admin USER ggadmin PASSWORD *****
```

Adminclient gets the real username and password from the wallet in the \$HOME directory of the user executing the script and use those credentials to connect to the service.

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 9-36 OBEY command file

```
ALTER CREDENTIALSTORE ADD USER c##ggadmin@cdb1 ALIAS cggwest DOMAIN
OracleGoldenGate PASSWORD ggadmin
DBLOGIN USERIDALIAS cggwest
ALTER CREDENTIALSTORE ADD USER ggadmin@pdbwest ALIAS ggwest DOMAIN
OracleGoldenGate PASSWORD Welcome20GG
```

```
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.

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.

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 [#unique_986](#)

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.

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 9-37 Parameter substitution on Windows

```
C:\> set EXTFILE=C:\ggs\extfile
C:\> set TABNAME=PROD.ACCOUNTS
C:\> replicat paramfile c:\ggs\dirprm\parmfl
```

Example 9-38 Parameter substitution on UNIX (korn shell)

```
$ EXTFILE=/ggs/extfile
$ export EXTFILE
$ TABNAME=PROD.ACCOUNTS
$ export TABNAME
$ replicat paramfile ggs/dirprm/parmfl
```

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](#).

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.

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 - 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.

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')
```

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

Character	Description
%	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
,	Comma
' '	Single quotes
" "	Double quotes
'	Accent mark (Diacritical mark)
.	Period
<	Less-than symbol (or beginning angle bracket)
>	Greater-than symbol (or ending angle bracket)

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 TARGETDB 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 pdbwest 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 pdbwest  
TABLE appraisal.sal*;
```

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_TEST*n* 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.

Performing Administrative Operations

This chapter contains instructions for making changes to applications, systems, and Oracle GoldenGate while the replication environment is active and processing data changes.

Shutting Down the System

When shutting down a system for maintenance and other procedures that affect Oracle GoldenGate, follow these steps to make certain that Extract has processed all of the transaction log records. Otherwise, you might lose synchronization data.

1. Stop all application and database activity that generates transactions that are processed by Oracle GoldenGate.
2. Run Admin Client and connect to the deployment using the CONNECT command.

3. Issue the `SEND EXTRACT` command with the `LOGEND` option. This command queries the Extract process to determine whether or not it is finished processing the records in the data source.

```
SEND EXTRACT group LOGEND
```

4. Continue issuing the command until it returns a YES status. At that point, all transaction log data has been processed, and you can safely shut down Oracle GoldenGate and the system.

Changing Database Attributes

This section addresses administrative operations that are performed on database tables and structures.

Changing Database Metadata

This procedure is required to prevent Replicat errors when changing the following metadata of the source database:

- Database character set
- National character set
- Locale
- Timezone
- Object name case-sensitivity

If these changes are made without performing this procedure, the following error occurs:

```
2013-05-26 20:10:09 ERROR   OGG-05500  Detected database metadata mismatch
between current trail file ./dirdat/_p/v1000000003 and the previous sequence.
*DBTIMEZONE: [GMT]/[UTC].
```

This procedure stops Extract, and then creates a new trail file. The new database metadata is included in this new file with the transactions that started after the change.

1. Stop transaction activity on the source database. Do not make the metadata change to the database yet.
2. On the source system, issue the `SEND EXTRACT` command with the `LOGEND` option until it shows there is no more redo data to capture.

```
SEND EXTRACT group LOGEND
```

3. Stop Extract.

```
STOP EXTRACT group
```

4. On each target system, issue the `SEND REPLICAT` command with the `STATUS` option until it shows a status of "At EOF" to indicate that it finished processing all of the data in the trail. This must be done on all target systems until all Replicat processes return "At EOF."

```
SEND REPLICAT group STATUS
```

5. Stop the data pumps and Replicat.

```
STOP EXTRACT group  
STOP REPLICAT group
```

6. Change the database metadata.

7. On the source system, issue the ALTER EXTRACT command with the ETROLLOVER option for the primary Extract to roll over the local trail to the start of a new file.

```
ALTER EXTRACT group, ETROLLOVER
```

8. Start Extract.

```
START EXTRACT group
```

9. Reposition the Replicat processes to start at the new trail sequence number.

```
ALTER REPLICAT group, EXTSEQNO seqno, EXTRBA RBA
```

10. Start the Replicat processes.

```
START REPLICAT group
```

Adding Tables to the Oracle GoldenGate Configuration

This procedure assumes that the Oracle GoldenGate DDL support feature is not in use, or is not supported for, your database.

Note

For Oracle and MySQL databases, you can enable the DDL support feature of Oracle GoldenGate to automatically capture and apply the DDL that adds new tables, instead of using this procedure. See the appropriate instructions for your database in this documentation.

Review these steps before starting. The process varies slightly, depending on whether or not the new tables satisfy wildcards in the TABLE parameter, and whether or not names or data definitions must be mapped on the target.

Prerequisites for Adding Tables to the Oracle GoldenGate Configuration

- This procedure assumes that the source and target tables are either empty or contain identical (already synchronized) data.
- You may be using the DBLOGIN USERIDALIAS and ADD TRANDATA commands. Before starting this procedure, see for the proper usage of these commands for your database.

To Add a Table to the Oracle GoldenGate Configuration

1. Stop user access to the new tables.
2. *(If new tables do not satisfy a wildcard)* If you are adding numerous tables that do not satisfy a wildcard, make a copy of the Extract and Replicat parameter files, and then add

the new tables with TABLE and MAP statements. If you do not want to work with a copy, then edit the original parameter files after you are prompted to stop each process.

3. (If new tables satisfy wildcards) In the Extract and Replicat parameter files, make certain the WILDCARDRESOLVEparameter is not being used, unless it is set to the default of DYNAMIC.
4. (If new tables do not satisfy a wildcard) If the new tables *do not* satisfy a wildcard definition, stop Extract.

```
STOP EXTRACT group
```

5. Add the new tables to the source and target databases.
6. If required for the source database, issue the ADD TRANDATA command for the new tables. Before using ADD TRANDATA, issue the DBLOGIN USERIDALIAS command.
7. Depending on whether the source and target definitions are identical or different, use either ASSUMETARGETDEFS or SOURCEDEFS in the Replicat parameter file. If SOURCEDEFS is needed, you can specify the template with the DEF option of the MAP parameter.
8. To register the new source definitions or new MAP statements, stop and then start Replicat.

```
STOP REPLICAT group
START REPLICAT group
```

9. Start Extract, if applicable.

```
START EXTRACT group
```

10. Permit user access to the new tables.

Coordinating Table Attributes between Source and Target

Follow this procedure if you are changing an attribute of a source table that is in the Oracle GoldenGate configuration, such as adding or changing columns or partitions, or changing supplemental logging details (Oracle). It directs you how to make the same change to the target table without incurring replication latency.

Note

This procedure assumes that the Oracle GoldenGate DDL support feature is not in use, or is not supported for your database. For Oracle and MySQL databases, you can enable the DDL support feature of Oracle GoldenGate to propagate the DDL changes to the target, instead of using this procedure.

1. On the source and target systems, create a table, to be known as the *marker table*, that can be used for the purpose of generating a marker that denotes a stopping point in the transaction log. Just create two simple columns: one as a primary key and the other as a regular column. For example:

```
CREATE TABLE marker
(
  id int NOT NULL,
  column varchar(25) NOT NULL,
  PRIMARY KEY (id)
);
```

2. Insert a row into the marker table on both the source and target systems.

```
INSERT INTO marker VALUES (1, 1);  
COMMIT;
```

3. On the source system, run Admin Client.
4. Open the Extract parameter file for editing.

 Caution

Do not use the VIEW PARAMS or EDIT PARAMS command to view or edit an existing parameter file that is in a character set other than that of the local operating system (such as one where the CHARSET option was used to specify a different character set). View the parameter file from outside the command line interface if this is the case; otherwise, the contents may become corrupted..

5. Add the marker table to the Extract parameter file in a TABLE statement.

```
TABLE marker;
```

6. Save and close the parameter file.
7. Add the marker table to the TABLE statement of the data pump, if one is being used.
8. Stop the Extract and data pump processes, and then restart them immediately to prevent capture lag.

```
STOP EXTRACT group  
START EXTRACT group
```

9. On the target system, run Admin Client.
10. Open the Replicat parameter file for editing.

 Caution

Do not use the VIEW PARAMS or EDIT PARAMS command to view or edit an existing parameter file that is in a character set other than that of the local operating system (such as one where the CHARSET option was used to specify a different character set). View the parameter file from outside Admin Client if this is the case; otherwise, the contents may become corrupted.

11. Add the marker table to the Replicat parameter file in a MAP statement, and use the EVENTACTIONS parameter as shown to stop Replicat and ignore operations on the marker table.

```
MAP marker, TARGET marker, EVENTACTIONS (STOP, IGNORE);
```

12. Save and close the parameter file.
13. Stop, and then immediately restart, the Replicat process.

```
STOP REPLICAT group  
START REPLICAT group
```

14. When you are ready to change the table attributes for both source and target tables, stop all user activity on them.
15. On the source system, perform an UPDATE operation to the marker table as the only operation in the transaction.

```
UPDATE marker  
SET column=2,  
WHERE id=1;  
COMMIT;
```

16. On the target system, issue the following command until it shows that Replicat is stopped as a result of the EVENTACTIONS rule.

```
STATUS REPLICAT group
```

17. Perform the DDL on the source and target tables, but do not yet allow user activity.
18. Start Replicat.

```
START REPLICAT group
```

19. Allow user activity on the source and target tables.

Dropping and Recreating a Source Table

Dropping and recreating a source table requires caution when performed while Oracle GoldenGate is running.

1. Stop access to the table.
2. Allow Extract to process any remaining changes to that table from the transaction logs. To determine when Extract is finished, use the INFO EXTRACT command.

```
INFO EXTRACT group
```

3. Stop Extract.

```
STOP EXTRACT group
```

4. Drop and recreate the table.
5. If supported for this database, run the ADD TRANDATA command for the table.
6. If the recreate action changed the source table's definitions so that they are different from those of the target, run the DEFGEN utility for the source table to generate source definitions, and then replace the old definitions with the new definitions in the existing source definitions file on the target system.
7. Permit user access to the table.

Purging Archive Logs

An Oracle archive log can be purged safely once Extract's read and write checkpoints are past the end of that log. Extract does not write a transaction to a trail until it has been committed, so Extract must keep track of all open transactions. To do so, Extract requires access to the archive log where each open transaction started and all archive logs thereafter.

Extract reads the current archive log (the read checkpoint) for new transactions and also has a checkpoint (the recovery checkpoint) in the oldest archive log for which there is an uncommitted transaction.

Use the following command to determine Extract's checkpoint positions.

```
INFO EXTRACT group, SHOWCH
```

- The Input Checkpoint field shows where Extract began processing when it was started.
- The Recovery Checkpoint field shows the location of the oldest uncommitted transaction.
- The Next Checkpoint field shows the position in the redo log that Extract is reading.

- The Output Checkpoint field shows the position where Extract is writing.

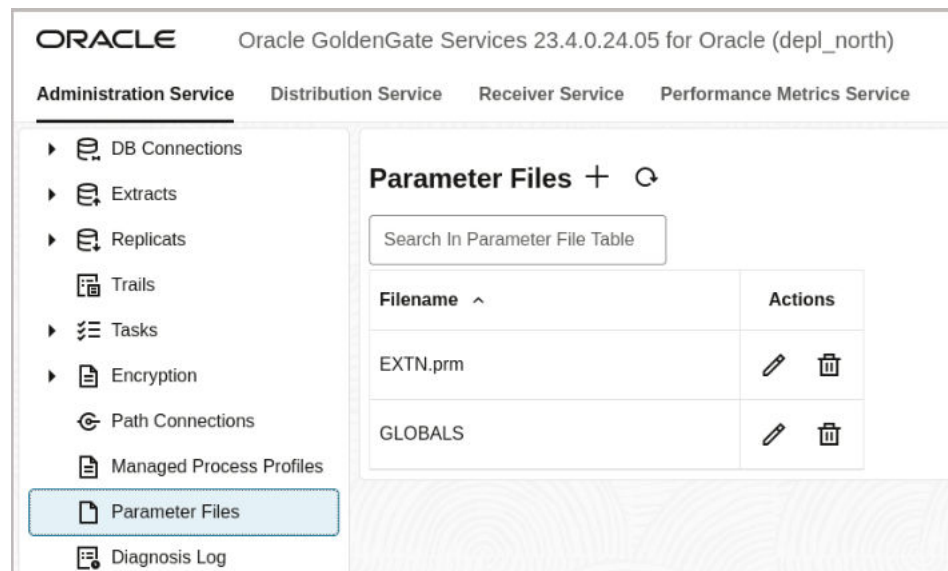
You can write a shell script that purges all archive logs no longer needed by Extract by capturing the sequence number listed under the Recovery Checkpoint field. All archive logs prior to that one can be safely deleted.

Reorganizing a DB2 Table (z/OS Platform)

When using IBM's REORG utility to reorganize a DB2 table that has compressed tablespaces, specify the KEEPDICTIONARY option if the table is being processed by Oracle GoldenGate. This prevents the REORG utility from recreating the compression dictionary, which would cause log data that was written prior to the change not to be decompressed and cause Extract to terminate abnormally. As an alternative, ensure that all of the changes for the table have been extracted by Oracle GoldenGate before doing the reorganization, or else truncate the table.

Access the Parameter Files

View and configure the Extract, Replicat, and Global parameter files from the **Parameter Files** options in the left navigation pane of the Administration Service.



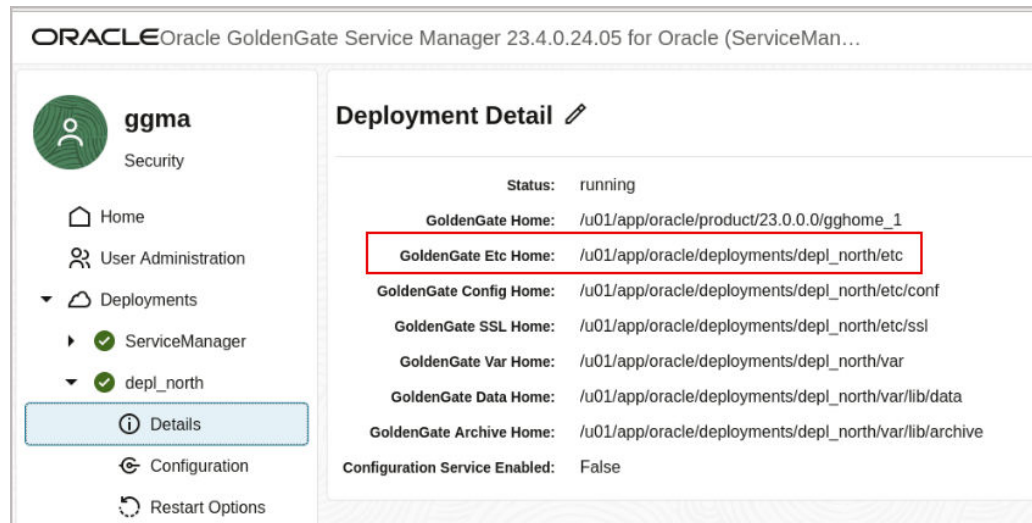
To use the different parameter file options:

1. Click the **Parameter Files** option.
A list of existing parameter files is displayed along with the GLOBALS parameter file.
2. 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 restart the Administration Service and any Extracts and Replicats.
3. Click + sign to add a parameter file.
4. Enter the file name and content.
5. Click **Submit**. The new parameter file is displayed in the list of parameter files.

View the Existing Parameter Files from the /ETC Directory

The actual location of the parameter files on the disk can be determined using the following steps:

1. Identify the GoldenGate Deployment ETC Home:
 - a. Go to Service Manager home page.
 - b. Click the deployment from the Deployments section for which you need to find the parameters file.
 - c. In the Deployment Detail page, view the path of the /etc home directory. The following image shows a sample of the /etc home directory location as seen from the Service Manager Deployment Detail page.



2. From the command prompt, navigate to the /config/ogg directory where the parameter file is located.

The following image shows how to navigate to your parameter file location:

```
>cd /u01/app/oracle/deployments/depl_north/etc/conf/ogg
>ls
EXTN.prm GLOBALS
```

Manage Auto Start and Auto Restart for Extract and Replicat Processes

Oracle GoldenGate Administration Service provides options to set up 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 Service or the Admin Client. To create a profile in the Administration Service, perform the following tasks:

1. Click Managed Process Profiles from the Administration Service navigation pane.
2. On the Managed Process Profiles screen, 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
Profile Name	Provides the name of the autostart and autorestart profile. You can select the default or custom options. If you have an existing profile, then you can select that profile also. If you select the Custom option, then 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. Mostly used for high availability scenarios.

Note

This option only appears if you are creating the profile for Extract or Replicat processes.

Auto Start	Enables autostart for the process.
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.

Switching Replicat from Non-Integrated Mode to Integrated Mode

Valid for Oracle only.

For more information about Replicat modes for an Oracle database, see [Types of Replicat](#).

This procedure switches an existing Replicat group from non-integrated to integrated mode.

Note

Do not configure the switch between Replicat modes to occur immediately after Extract recovers from a failure or is repositioned to a different location in the transaction log.

1. Back up the Oracle GoldenGate working directories.

2. While the Oracle GoldenGate processes continue to run in their current configuration, so that they keep up with current change activity, copy the Replicat parameter file to a new name.
3. Grant the appropriate role to the Replicat user and perform the required configuration steps to support your business applications in integrated Replicat mode.
4. Run the command line (Admin Client).
5. Stop Replicat.

```
STOP REPLICAT group
```

6. Log into the target database from GGSCI.

```
DBLOGIN USERIDALIAS alias
```

Where: *alias* is the alias of a user in the credential store who has the required user roles when using Oracle GoldenGate 23ai with Oracle AI Database 26ai and the privileges granted through Oracle `dbms_goldengate_auth.grant_admin_privilege` procedure when using Oracle GoldenGate 21c . See [Grant User Privileges for Oracle AI Database 26ai and Higher](#) and [Grant User Privileges for Oracle Database 21c and Lower](#) to know more.

7. Alter Replicat to integrated mode.

```
ALTER REPLICAT group, INTEGRATED
```

8. Replace the old parameter file with the new one, keeping the same name.
9. Start Replicat.

```
START REPLICAT group
```

10. Verify that Replicat is in integrated mode.

```
INFO REPLICAT group
```

When you start Replicat in integrated mode for the first time, the START command registers the Replicat group with the database and starts an inbound server to which Replicat attaches. When you convert a Replicat group to integrated mode, the use of the Oracle GoldenGate checkpoint table is discontinued and recovery information is maintained internally by the inbound server and by the checkpoint file going forward. You can retain the checkpoint table in the event that you decide to switch back to non-integrated mode.

Switching Replicat from Integrated Mode to Non-Integrated Mode

Valid for Oracle only. For more information about Replicat modes for an Oracle database, see [About Integrated Replicat](#).

You can, at any time, switch Replicat from integrated mode to non-integrated mode. This switch automatically unregisters the Replicat group from the target database, which removes the inbound server.

Note

Do not configure the switch between Replicat modes to occur immediately after Extract recovers from a failure or is repositioned to a different location in the transaction log.

1. Back up the Oracle GoldenGate working directories.
2. While the Oracle GoldenGate processes continue to run in their current configuration, so that they keep up with current change activity, copy the Replicat parameter file to a new name.
3. Grant the appropriate privileges to the Replicat user and perform the required configuration steps to support your business applications in non-integrated Replicat mode.
4. Connect to the Oracle GoldenGate deployment as a system user.
5. Log into the target database from the Admin Client.

```
DBLOGIN USERIDALIAS alias
```

Where: *alias* is the alias of a user in the credential store who has the required user roles when using Oracle GoldenGate 23ai with Oracle AI Database 26ai and the privileges granted through Oracle `dbms_goldengate_auth.grant_admin_privilege` procedure when using Oracle GoldenGate 21c . See [Grant User Privileges for Oracle AI Database 26ai and Higher](#) and [Grant User Privileges for Oracle Database 21c and Lower](#) to know more.

6. Create a checkpoint table in the target database for the non-integrated Replicat to store its recovery checkpoints. If a checkpoint table was previously associated with this Replicat group and still exists, you can omit this step. See [Add a Checkpoint Table](#) for more information about options for using a checkpoint table.

```
ADD CHECKPOINTTABLE [container.]table
```

7. Stop Replicat.

```
STOP REPLICAT group
```

8. Alter Replicat to non-integrated mode. For the CHECKPOINTTABLE argument, specify the checkpoint table that you created for this Replicat group.

```
ALTER REPLICAT group, NONINTEGRATED, CHECKPOINTTABLE [container.]table
```

9. Replace the old parameter file with the new one, keeping the same name.
10. Start Replicat.

```
START REPLICAT group
```

After issuing this command, wait until there is some activity on the source database so that the switchover can be completed. (Replicat waits until its internal high-water mark is exceeded before removing the status of "switching from integrated mode.")

11. Verify that Replicat switched to non-integrated mode.

INFO REPLICAT *group*

Switching Replicat to Coordinated Mode

Valid for all database types supported by Oracle GoldenGate.

This procedure upgrades a regular Replicat configuration (non-coordinated) to a coordinated configuration. This procedure assumes you are replacing a configuration that partitions data across multiple Extract and Replicat processes with a configuration that uses one Extract and one coordinated Replicat. The coordinated Replicat replaces the need for using multiple Replicat processes. A coordinated Replicat requires only one trail, so there is no need for multiple Extract processes.

Procedure Overview

This procedure makes use of the `EVENTACTIONS` parameter with a `STOP` action, which enables all of the Replicat processes to stop at the same point in the trail. The `EVENTACTIONS` action is triggered by a transaction that contains an `INSERT` to a dummy table. The `INSERT` causes each process to finish processing everything up to, and including, the event transaction and then stop cleanly. An additional event action of `IGNORE` is specified for Replicat to prevent the multiple Replicat processes from attempting to insert the same record to the target. The result of this procedure is that all processes stop at the same point in the data stream: after completing the `INSERT` transaction to the dummy table.

After the processes stop, you move all of the `TABLE` statements to one primary Extract group. You move all of the `MAP` statements to a new coordinated Replicat group that reads the trail associated with the retained Extract. After all of the `MAP` statements are in one parameter file, you edit them to add the thread specifications to support a coordinated Replicat. (This can be done ahead of time.) Then you drop the Replicat group and add it back in coordinated mode with the same name.

Performing the Switch to Coordinated Replicat

Note

Do not create the Replicat group until prompted by these instructions.

1. Back up the current parameter files of all of the Extract groups and Replicat groups. You will be editing them.
2. Create a working directory outside the Oracle GoldenGate directory. You will use this directory to create and stage new versions of the parameter files. If needed, you can create a working directory on the source and target systems.
3. In the working directory, create a parameter file for a coordinated Replicat. Copy the `MAP` parameters from the active parameter files of all of the Replicat groups to this parameter file, and then add the thread specifications and any other parameters that support your required coordinated Replicat configuration.
4. If using multiple primary Extract groups, select one to keep, and then save a copy of its current parameter file to the working directory.

5. Copy all of the TABLE statements from the other Extract groups to the new parameter file of the primary Extract that you are keeping.
6. Copy all of the TABLE statements from the other Extract to the copied parameter file of the retained Extract.
7. In the source database, create a simple dummy table on which a simple INSERT statement can be performed. For this procedure, the name `schema.event` is used.
8. Create the same table on the target system, to avoid the need for additional configuration parameters.
9. Edit the active parameter files (not the copies) of all primary and data-pump Extract groups to add the following EVENTACTIONS parameter to each one.

```
TABLE schema.event, EVENTACTIONS(STOP);
```

10. Edit the active parameter files (not the copies) of all of the Replicat groups to add the following EVENTACTIONS parameter to each one.

```
MAP schema.event, TARGET schema.event, EVENTACTIONS(IGNORE, STOP);
```

11. Stop the Oracle GoldenGate processes gracefully in the following order:
 - Stop all Replicat processes.
 - Stop all Extract processes.
12. Restart the Oracle GoldenGate processes in the following order so that the EVENTACTIONS parameters take effect:
 - Start all Extract processes.
 - Start all Replicat processes.
13. On the source system, issue a transaction on the `schema.event` table that contains one INSERT statement. Make certain to commit the transaction.
14. In Admin Client, issue the STATUS command for all of the primary Extract on the source system, and issue the same command for all of the Replicat processes on the target system, until the commands show that all of the processes are STOPPED.

```
STATUS EXTRACT *
STATUS REPLICAT *
```

15. Replace the active parameter files of the Extract that you retained with the new parameter files from the working directory.
16. Delete the unneeded Extract and its parameter files.
17. Log into the target database by using the DBLOGIN USERIDALIAS command.
18. Delete all of the Replicat groups and their active parameter files.
19. Copy or move the new coordinated Replicat parameter file from the working directory to the Oracle GoldenGate directory.
20. In Admin Client, issue the INFO EXTRACT command and make note of its write checkpoint position in the trail.

```
INFO EXTRACT extract_name, DETAIL
```

21. Add a new coordinated Replicat group with the following parameters.

```
ADD REPLICAT group, EXTTRAIL trail, EXTSEQNO sequence_number, EXTRBA rba,
COORDINATED MAXTHREADS number
```

Where:

- *group* is the name of the coordinated Replicat group. The name must match that of the new parameter file created for this group.
- *EXTTRAIL trail* is the name of the trail that the Extract writes to.
- *EXTSEQNO sequence_number* is the sequence number of the trail as shown in the write checkpoint returned by the `INFO EXTRACT` that you issued for the Extract.
- *EXTRBA rba* is the relative byte address in the trail as shown in the write checkpoint returned by `INFO EXTRACT`. Together, these position Replicat to resume processing at the correct point in the trail.
- *MAXTHREADS* number specifies the maximum number of threads allowed for this group. This value should be appropriate for the number of threads that are specified in the parameter file.

22. Start the primary Extract group.

23. Start the coordinated Replicat group.

Administering a Coordinated Replicat Configuration

This section contains instructions for coordinating threads and re-partitioning the workload among new or different threads. A coordinated Replicat should be stopped cleanly with the `STOP REPLICAT` command before making modifications to the partition specifications in `THREAD` or `THREADRANGE` clauses of the `MAP` statements. A clean stop ensures that all of the threads, which may be at different locations in the trail at any given point, all finish their work and arrive at a common trail location.

At startup, Replicat issues an error and abends if it detects that the last shutdown was not clean and the partitioning in the `MAP` statements was changed to contain a different number of threads (threads were added or removed). However, if the same threads are kept in the parameter file but simply rearranged among different `MAP` statements, Replicat issues a warning but does not abend. This can result in missing or duplicate records, because there is no way to ensure continuity of the thread-to-workload allocations from the previous run.

The following is an example of this condition.

Following is the original partitioning scheme:

```
MAP source, target, THREADRANGE(1-5);  
MAP source1, target1, THREADRANGE(6-10);
```

The following re-partitioning of the original scheme produces only a warning:

```
MAP source, target, THREADRANGE(1-4);  
MAP source1, target1, THREADRANGE(5-10);
```

This section provides instructions for cleanly shutting down Replicat before performing a re-partitioning, as well as instructions for attempting to recover Replicat continuity when a re-partitioning is performed after an unclean shutdown.

The following tasks can be performed for a Replicat group in coordinated mode.

Performing a Planned Re-partitioning of the Workload

A planned re-partitioning is when Replicat is allowed to shut down cleanly before it is started again with a new parameter file that contains updated thread partitioning. A clean shutdown enables all of the threads to arrive at a common checkpoint position in the trail. At that point, the new partitioning scheme can be applied in the next run.

1. Run Admin Client.
2. Stop Replicat.
`STOP REPLICAT group`
3. Open the parameter file for editing.
`EDIT PARAMS group`
4. Make the required changes to the `THREAD` or `THREADRANGE` specifications in the `MAP` statements.
5. Save and close the parameter file.
6. Start Replicat.
`START REPLICAT group`

Recovering Replicat After an Unplanned Re-partitioning

An *unplanned re-partitioning* is when Replicat is not allowed to shut down cleanly before it is started again with a new parameter file that contains updated thread partitioning. In this scenario, some or all of the old threads were not able to finish their work and arrive at a common checkpoint. Upon restart, the coordinator thread attempts to apply the old partitioning scheme, and Replicat abends with an error. You can recover the coordinated Replicat group from this condition in one of the following ways:

- Use the auto-saved copy of the parameter file
- Reprocess from the low watermark with `HANDLECOLLISIONS`

Reprocessing From the Low Watermark with `HANDLECOLLISIONS`

In this procedure, you reposition all of the threads to the *low watermark* position. This is the earliest checkpoint position performed among all of the threads. To state it another way, the low watermark position is the last record processed by the slowest thread before the unclean stop. When you start Replicat, the threads reprocess the operations that they were processing before Replicat stopped, and the `HANDLECOLLISIONS` parameter handles any duplicate-record and missing-record errors that occur as the faster threads reprocess operations that they applied before the unclean stop.

1. Add the `HANDLECOLLISIONS` parameter to the Replicat parameter file. It is not necessary to use any `THREADS` options.
2. Issue the `INFO REPLICAT` command for the Replicat group as a whole (the coordinator thread). Make a record of the RBA of the checkpoint. This is the *low watermark* value. This output also shows you the active thread IDs under the `Group Name` column. Make a record of these, as well.

`INFO REPLICAT group`

For example:

```
OGG(slc03jgo) 3> info ra detail REPLICAT RA      Last Started
2013-05-01 14:15   Status ABENDEDCOORDINATED
Coordinator                               MAXTHREADS 15Checkpoint Lag
00:00:00 (updated 00:00:07 ago)Process ID      11445Log Read
Checkpoint File ./dirdat/withMaxTransOp/bg000000001
2013-05-02 07:49:45.975662 RBA 44704Lowest Log BSN value: (requires
database login)Active Threads: ID  Group Name PID   Status   Lag at Chkpt
```

```

Time Since Chkpt1  RA001      11454 ABENDED  00:00:00      00:00:01      2
RA002      11455 ABENDED  00:00:00      00:00:04      3  RA003      11456
ABENDED  00:00:00      00:00:01      5  RA005      11457 ABENDED
00:00:00      00:00:02      6  RA006      11458 ABENDED  00:00:00
00:00:04      7  RA007      11459 ABENDED  00:00:00      00:00:04

```

3. Issue the `INFO REPLICAT` command for each processing thread ID and record the RBA position of each thread. Make a note of the *highest* RBA. This is the *high watermark* of the Replicat group.

```
INFO REPLICAT threadID
```

For example:

```

info ra002
REPLICAT RA002      Last Started 2013-05-01 14:15      Status
ABENDEDCOORDINATED      Replicat Thread      Thread
2Checkpoint Lag      00:00:00 (updated 00:00:06 ago)Process ID
11455
Log Read Checkpoint  File ./dirdat/withMaxTransOp/
bg000000001      2013-05-02 07:49:15.837271  RBA 45603

```

4. Issue the `ALTER REPLICAT` command for the coordinator thread (Replicat as a whole, without any thread ID) and position to the *low watermark* RBA that you recorded.

```
ALTER REPLICAT group EXTRBA low_watermark_rba
```

5. Start Replicat.

```
START REPLICAT group
```

6. Issue the basic `INFO REPLICAT` command until it shows an RBA that is higher than the *high watermark* that you recorded. `HANDLECOLLISIONS` handles any collisions that occur due to previously applied transactions.

```
INFO REPLICAT group
```

7. Stop Replicat.

```
STOP REPLICAT group
```

8. Remove or comment out the `HANDLECOLLISIONS` parameter.

9. Start Replicat.

```
START REPLICAT group
```

Using the Auto-Saved Parameter File

A copy of the original parameter file is saved whenever the parameter file is edited before shutting down Replicat cleanly. You can revert to this parameter file and then resynchronize the threads so that they all catch up to the thread that had the most recent checkpoint position. Once the threads are synchronized, you can switch to the new parameter file and then start Replicat.

1. Save the new parameter file to a different name, and then rename the saved original parameter file to the correct name (same as the group name). The saved parameter file has a `.backup` suffix and is stored in the `dirprm` subdirectory of the Oracle GoldenGate installation directory.

2. Issue the following command to synchronize the Replicat threads to the maximum checkpoint position. This command automatically starts Replicat and executes the threads until they reach the maximum checkpoint position.

```
SYNCHRONIZE REPLICAT group
```

3. Issue the STATUS REPLICAT command until it shows that Replicat stopped cleanly.

```
STATUS REPLICAT group
```

4. Save the original parameter file to a different name, and then rename the new parameter file to the group name.

5. Start Replicat.

```
START REPLICAT group
```

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.
<code>#macro_name</code>	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>. 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>. 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 in the <i>Parameters and Functions Reference for Oracle GoldenGate</i>. Macro and parameter names are not case-sensitive. Macro or parameter names within quotation marks are ignored.
PARAMS (<code>#p1</code> , <code>#p2</code>)	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.
<code>macro_body</code>	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. - Simple parameter statements, as in: <pre>COL1 = COL2</pre> - Complex parameter statements with parameter substitution as in: <pre>MAP #o.#t, TARGET #o.#t, KEYCOLS (#k), COLMAP (USEDEFAULTS);</pre> - Invocations of other macros, as in: <pre>#colmap (COL1, #sourcecol)</pre>
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
<i>target</i> =	Optional. Specifies the target to which the results of the macro are assigned or mapped. For example, <i>target</i> 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. DATECOL1 = #make_date (YR1, M01, DAY1) Without a target, the syntax to call <code>#make_date</code> is: #make_date (YR1, M01, DAY1)
<i>macro_name</i>	The name of the macro that is being called, for example: <code>#make_date</code> .

Table 9-19 (Cont.) Syntax Elements for Calling a Macro

Argument	Description
<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 MACRO 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 MACRO 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, {DEPTN01, DEPTN02, DEPTN03})`. 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 9-39 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, DEPTN01)
```

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, {DEPTN01, DEPTN02})
```

The DEPTN01 and DEPTN02 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 9-40 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
(
targcol1 = sourcecol1,
datecol1 = #make_date(YR1, M01, DAY1),
datecol2 = #make_date(YR2, M02, DAY2)
);
```

The macro expands as follows:

```
MAP sales.acct_tab, TARGET sales.account,
COLMAP
(
targcol1 = sourcecol1,
datecol1 = @DATE ('YYYY-MM-DD', 'CC', @IF (YR1 < 50, 20, 19), 'YY', YR1, 'MM',
```

```

M01, 'DD', DAY1),
datecol2 = @DATE ('YYYY-MM-DD', 'CC', @IF (YR2 < 50, 20, 19), 'YY', YR2, 'MM',
M02, '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.targtab;

GETINSERTS
GETUPDATES
GETDELETES
INSERTDELETES
MAP owner.srctab2, TARGET owner.targtab2;

```

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 ERDCALLBACK 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 9-41 Example of Base Syntax, UNIX

```
CUSEREXIT eruserexit.so MyUserExit
```

Example 9-42 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.

Table 9-21 (Cont.) column_value_mode_matrix Meanings

Data Type	EXIT_FN_RAW_FORMAT	EXIT_FN_CHAR_FORMAT	EXIT_FN_CNVTED_SESS_FORMAT
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.
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

Monitor and Tune

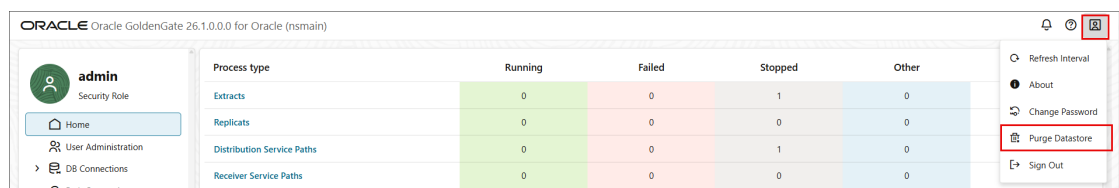
Learn about different techniques available with Oracle GoldenGate to carry out performance monitoring and tuning activities.

Monitor

Learn about monitoring Oracle GoldenGate processes for performance and error handling.

Purge Datastore

You can change the datastore retention and purge it from the top-right navigation menu:



Clicking the Purge Datastore option opens the Purge Datastore dialog box. You can enter the retention days to set the new period from 1 to 365 days, then **Execute** to activate the purge. The details of the purge are also displayed.

Purge Datastore

Retention Days:
15

Cancel Execute

Monitor Extracts

To view the performance data of Extracts, click Extracts from the left navigation pane. The Extract page is displayed. You can view the following details for Extracts on the Extracts dashboard page:

- Process Name: Extract process name
- Process ID: Extract process ID.

- Status: Status of Extract depending on whether the Extract is Running, Stopped, or Abended.
- Heartbeat Lag: Value of the heartbeat lag for the Extract.
- Heartbeat Age: Value of the duration for which the heartbeat data has been collected for the Extract.
- Last Start Time: Time when the Extract was last started.

If you click a specific Extract, you can also view the performance data for the Extract process for the following:

- Process Performance
- Thread Performance
- Status and Configuration
- Trail Files
- Database Statistics
- Procedure Statistics
- Cache Statistics
- Queue Statistics
- Bounded Recovery

Monitor Replicats

To view the performance data of Replicats, click Replicats from the left navigation pane of the Performance Metrics Service. On the Replicats page, you can see the general status of all Replicats, along with general performance data in a tabular format.

- Process Name: Extract process name
- Process ID: Extract process ID.
- Status: Status of Extract depending on whether the Extract is Running, Stopped, or Abended.
- Heartbeat Lag: Value of the heartbeat lag for the Extract.
- Heartbeat Age: Value of the duration for which the heartbeat data has been collected for the Extract.
- Last Start Time: Time when the Extract was last started.

Click the Replicat for which you want to deep-dive into performance and other monitoring details. This displays a graphical presentation of performance information for that Replicat. It would include details for:

- Process Performance
- Thread Performance
- Status and Configuration
- Trail Files
- Database Statistics
- Procedure Statistics
- Bounded Recovery

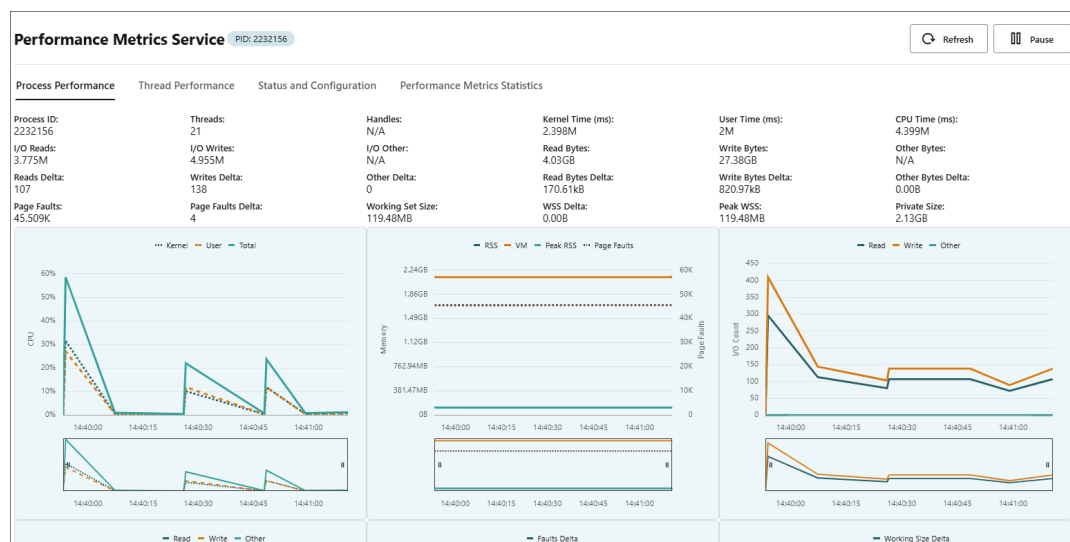
Monitor Microservices

From the **Services** option in the left-navigation pane, click any of the Microservices: **Administration Service**, **Distribution Service**, **Receiver Service**, or the **Performance Metrics Service**. The **Metrics** page is displayed, which displays the metrics for **Process Performance**, **Thread Performance**, **Status and Configuration**, and **Performance Metrics Statistics** (Performance Metrics Service only).

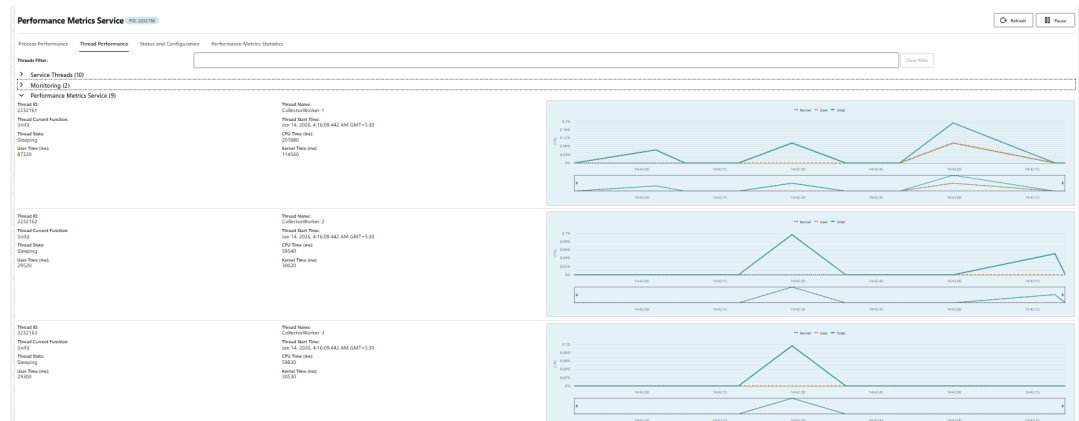
Note

The diagrams in this topic are a sample from the Performance Metrics Service monitoring information. The other microservices also display similar information.

- **Process Performance** tab displays process related details including the process ID, number of threads, CPU time, I/O Read, I/O Writes and many others.



- **Thread Performance** tab thread level data at the **Service Thread** level, **Monitoring** level to show the threads that are being monitored, and the **Performance Metrics Service** thread monitoring.



- **Status and Configuration** tab displays the process messages and process status changes.

Performance Metrics Service PID: 2232156

Process Performance Thread Performance **Status and Configuration** Performance Metrics Statistics

Process Basic Information

Process ID: 2232156
 Process Mode: Not Applicable
 Process Start Time: Jan 14, 2026, 4:16:09 AM GMT+5:30

Process Type: PMServer
 Process Status: Running

Process Messages Process Status Changes

Search In Process Message Table [X] Refresh

Code	Date	Process	Severity	Message
OGG-08100	Jan 13, 2026, 4:53:17 AM GMT+5:30	PMSRVR	Success	Profile 'Trace' added.

- **Performance Metrics Statistics** tab displays data for the Performance Metrics Service, including packets received, bytes of data received, database type and others.

Performance Metrics Service PID: 2232156

Process Performance Thread Performance Status and Configuration **Performance Metrics Statistics**

Packets Received: 1636691	Bytes Received: 512565872	Receive Errors: 0
Max Packet Size: 8640	Min Packet Size: 173	Sequence Errors: 0
Missing Packets: 0	Invalid Packets: 0	Recv Buffer Size: 425984
Database Type: BDB		

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\).

Recover from Datastore Corruption

The Performance Metrics Service collects and manages operational metrics related to the health and performance of replication processes. The dirbdb directory stores essential checkpoint information. If contents of this directory are missing or corrupted, then the Performance Metric Service may not be able to collect performance data.

To resolve this issue, you can **disable** the Performance Metrics Service from the Service Manager or the Admin Client, and then enable it again. This will recreate corrupted BDB files and resolve the issue.

Using Automatic Workload Repository (AWR) Reports for Oracle Database

Automatic Workload Repository (AWR) is a good starting point for identifying general database performance issues, which can provide initial indicators to help locate problems with Extract or Replicat processes. Using AWR, you can easily determine if the bottlenecks are inside or outside of the database.

Starting with Oracle Database 23ai, AWR queries and reports are simplified and enhanced to present the data in an easy to understand view of the replication process. With these report views, it would be easier to troubleshoot replication performance issues by categorizing them as one of the following:

- Workload issue
- Database-side misconfiguration issue, such as slow replication SQL due to lack of indexes or wrong database parameter settings
- Performance bottleneck issue at the database side or in the Oracle GoldenGate processes outside of the database

The replication sections of the enhanced AWR reports include the following features:

- A more complete Replication System Resource Usage section, which shows the system resource usage for all Oracle GoldenGate replication processes, whether they be foreground or background, and presented for each Extract and Replicat process.

The Replication section also contains information for the XStream IN/OUT interface. For more information about XStream, see *Oracle Database XStream Guide*.

- A separate section for replication related Top SQL statistics to make it easier to identify replication related SQL performance issues. The number of Top SQL shown is a fraction of the Top SQL shown for the database.
- A separate section for top wait events for replication related processes to easily troubleshoot replication related performance problems.
- Reorganized replication related sections to present the replication statistics organized by individual Extract and different Replicat types.
- Customized information for Replicat and Extract processes.

Replication System Resource Usage

Oracle GoldenGate replication process name, process type and number of sessions of its sub-components are displayed in the metrics. The performance statistic are aggregated by the functionality of process sub-components, group by process name. This allows the ability to monitor resource usage at per Extract and Replicat, with detailed information of all its sub-components.

Replication System Resource Usage

- System resource usage of GoldenGate/XStream processes aggregated by Replication Group, Session Type, Process Type and Session Module
- Ordered by Replication Group in ascending order, CPU Time in descending order, followed by Session Type and Session Module in ascending order

Replication Group	Number of Sessions	Session Type	Process Type	Session Module	First Logon	CPU Time(sec)	User IO Wait Time(sec)	System IO Wait Time(sec)	PDB Name
EXT1A	1	Logminer Builder	Integrated Extract	GoldenGate	22-Nov-21 10:07:55	2.20	0.10	0.00	CDB1_PDB1
EXT1A	2	Logminer Preparer	Integrated Extract	GoldenGate	22-Nov-21 10:07:56	0.29	0.00	0.00	CDB1_PDB1
EXT1A	1	Logminer Reader	Integrated Extract	GoldenGate	22-Nov-21 10:07:58	0.26	0.00	0.02	CDB1_PDB1
EXT1A	3	Metadata	Integrated Extract	GoldenGate	22-Nov-21 10:07:42	0.18	0.00	0.00	CDB1_PDB1
EXT1A	1	Capture	Integrated Extract	GoldenGate	22-Nov-21 10:07:52	0.10	0.00	0.00	CDB1_PDB1
EXT1A	1	Logminer Merger	Integrated Extract	GoldenGate	22-Nov-21 10:07:53	0.05	0.00	0.00	CDB1_PDB1
REP3F	1	Applier	Classic Replicat	GoldenGate	22-Nov-21 10:07:36	1.14	0.01	0.00	CDB1_PDB1
REP3G	1	Thread	Coordinated Replicat	GoldenGate	22-Nov-21 10:07:49	1.13	0.02	0.00	CDB1_PDB1
REP3G	1	Apply	Coordinated Replicat	GoldenGate	22-Nov-21 10:07:36	0.00	0.00	0.00	CDB1_PDB1
REP3H	4	Applier	Parallel Replicat	GoldenGate	22-Nov-21 10:07:49	1.33	0.00	0.00	CDB1_PDB1
REP3H	2	Mapper	Parallel Replicat	GoldenGate	22-Nov-21 10:07:49	0.06	0.00	0.00	CDB1_PDB1
REP3H	1	Main	Parallel Replicat	GoldenGate	22-Nov-21 10:07:36	0.00	0.00	0.00	CDB1_PDB1
REP4I	4	Apply Receiver	Parallel Integrated Replicat	GoldenGate	22-Nov-21 10:07:49	1.35	0.00	0.00	CDB1_PDB1
REP4I	4	Mapper	Parallel Integrated Replicat	GoldenGate	22-Nov-21 10:07:49	0.55	0.00	0.00	CDB1_PDB1
REP4I	1	Main	Parallel Integrated Replicat	GoldenGate	22-Nov-21 10:07:36	0.00	0.00	0.00	CDB1_PDB1
REP4J	1	Apply Receiver	Integrated Replicat	GoldenGate	22-Nov-21 10:07:37	0.81	0.02	0.00	CDB1_PDB1
REP4J	1	Apply Reader	Integrated Replicat	GoldenGate	22-Nov-21 10:07:46	0.44	0.00	0.00	CDB1_PDB1
REP4J	5	Apply Server	Integrated Replicat	GoldenGate	22-Nov-21 10:07:49	0.18	0.00	0.00	CDB1_PDB1
REP4J	1	Apply Coordinator	Integrated Replicat	GoldenGate	22-Nov-21 10:07:44	0.00	0.00	0.00	CDB1_PDB1
X1ACAP	1	Capture	XStream OUT	XStream	22-Nov-21 10:11:57	1.60	0.05	0.02	CDB\$ROOT
X2BCAP	1	Capture	XStream OUT	XStream	22-Nov-21 10:11:56	1.66	0.04	0.02	CDB\$ROOT
XIN3C	5	Apply Server	XStream IN	XStream	22-Nov-21 10:11:09	0.91	0.00	0.00	CDB1_PDB1
XIN3C	1	Apply Reader	XStream IN	XStream	22-Nov-21 10:11:09	0.45	0.04	0.00	CDB1_PDB1
XIN3C	1	Propagation Receiver	XStream IN	XStream	22-Nov-21 10:16:55	0.14	0.00	0.00	CDB1_PDB1
XIN3C	1	Apply Coordinator	XStream IN	XStream	22-Nov-21 10:11:08	0.00	0.00	0.00	CDB1_PDB1
XIN4D	5	Apply Server	XStream IN	XStream	22-Nov-21 10:11:12	1.00	0.00	0.00	CDB1_PDB1
XIN4D	1	Apply Reader	XStream IN	XStream	22-Nov-21 10:11:12	0.46	0.09	0.00	CDB1_PDB1
XIN4D	1	Propagation Receiver	XStream IN	XStream	22-Nov-21 10:17:02	0.14	0.00	0.00	CDB1_PDB1
XIN4D	1	Apply Coordinator	XStream IN	XStream	22-Nov-21 10:11:11	0.00	0.00	0.00	CDB1_PDB1
XOUT1A	1	Apply Reader	XStream OUT	XStream	22-Nov-21 10:11:26	0.25	0.00	0.00	CDB\$ROOT
XOUT1A	2	Apply Server	XStream OUT	XStream	22-Nov-21 10:11:26	0.18	0.00	0.00	CDB\$ROOT
XOUT1A	1	Propagation SendRcv	XStream OUT	XStream	22-Nov-21 10:11:57	0.11	0.00	0.00	CDB\$ROOT

Replication Top Wait Events

Replication foreground and background processes wait events are shown, aggregated by event type, and Oracle GoldenGate SQL Module.

Replication Foreground Wait Events

- s - second, ms - millisecond, us - microsecond, ns - nanosecond
- Events with Total Wait Time (sec) >= .01 are shown
- %Timeouts: value of 0 indicates value was < .5%. Value of null is truly 0
- Ordered by Total Wait Time in descending order, followed by number of Waits in descending order desc (idle events last)

Event	Event ID	Session Module	Waits	%Time -outs	Total Wait Time (sec)	Avg wait	Waits /txn	% DB time
REPL Capture/Apply: miscellaneous	2721331920	GoldenGate	267	14	1,151	4310.25ms	0.04	42.01
SQL*Net more data from client	3530226808	XStream	3,276		49	14.88ms	0.44	1.78
latch: shared pool	2696347763	GoldenGate	9,043		46	5.06ms	1.20	1.67
cursor: pin S wait on X	1729366244	GoldenGate	38		23	603.77ms	0.01	0.84
SQL*Net more data to client	554161347	XStream	2,594		23	8.77ms	0.34	0.83
free buffer waits	2701153470	GoldenGate	1,506		19	12.80ms	0.20	0.70
write complete waits	4229542060	GoldenGate	67		11	165.23ms	0.01	0.40
library cache: mutex X	1646780882	GoldenGate	1,269		10	7.70ms	0.17	0.36
enq: CR - block range reuse ckpt	2576559565	GoldenGate	21		0	10.41ms	0.00	0.01

Replication Background Wait Events

- Only top 20 events with Total Wait Time (sec) >= .01 are shown
- %Timeouts: value of 0 indicates value was < .5%. Value of null is truly 0
- Ordered by Total Wait Time in descending order, followed by number of Waits in descending order (idle events last)

Event	Event ID	Session Module	Waits	%Time -outs	Total Wait Time (sec)	Avg wait	Waits /txn	% bg time
db file sequential read	2652584166	GoldenGate	33,891	0	78	2.31ms	4.50	3.93
db file sequential read	2652584166	XStream	33,891	0	78	2.31ms	4.50	3.93
latch: shared pool	2696347763	GoldenGate	7,076	0	60	8.42ms	0.94	2.99
latch: shared pool	2696347763	XStream	7,076	0	60	8.42ms	0.94	2.99
enq: MN - contention	1489162918	XStream	586	0	32	54.44ms	0.08	1.60
enq: MN - contention	1489162918	GoldenGate	586	0	32	54.44ms	0.08	1.60
oracle thread bootstrap	4097944222	GoldenGate	58	0	24	412.66ms	0.01	1.20

Replication Top SQLs

SQLs that are executed by replication processes, displayed in different sections ordered by Elapsed Time, CPU Time and Execution are presented in the Top SQL report. Replication process name is added for identifying the process that is executing the SQL.

Here are examples of the top SQL reports.

Replication SQL ordered by Elapsed Time

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code for GoldenGate and XStream only
- % Total DB Time is the Elapsed Time of the SQL statement divided into the Total Database Time multiplied by 100
- %Total - Elapsed Time as a percentage of Total DB time
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Ordered by Elapsed Time in seconds in descending order.

Elapsed Time (sec)	Executions	Elapsed Time per Exec (sec)	%Total	%CPU	%IO	SQL ID	SQL Module	Process Name	SQL Text	PDB Name
1.78	7	0.25	3.13	51.23	0.00	d4rmz5j1x44b	GoldenGate	REP4I	SELECT DISTINCT c.name, c.col#...	CDB1_PDB1
0.66	2,001	0.00	1.15	92.42	0.08	38v3jc2wh3zjf	XStream	XIN4D	DELETE /*+ restrict_all_ref_c...	CDB1_PDB1
0.62	2,001	0.00	1.08	50.64	0.00	cdtx7d145sq4x	GoldenGate	REP3F	INSERT INTO "TKGGU1"."REP3F" (...)	CDB1_PDB1
0.61	2,001	0.00	1.07	54.76	0.00	bnnmm4zat1nn8	GoldenGate	REP3G	INSERT INTO "TKGGU1"."REP3G" (...)	CDB1_PDB1
0.59	2,001	0.00	1.03	92.38	0.09	cys9jimbq3gnr	XStream	XIN3C	DELETE /*+ restrict_all_ref_c...	CDB1_PDB1
0.43	2,001	0.00	0.76	83.04	0.00	bp7jxx1uwn0m	GoldenGate	REP4I	INSERT /*+ restrict_all_ref_c...	CDB1_PDB1
0.42	2,001	0.00	0.74	71.17	0.12	ac4shy6z3kd2b	GoldenGate	REP3H	INSERT INTO "TKGGU1"."REP3H" (...)	CDB1_PDB1
0.41	7	0.06	0.72	57.35	0.00	bn9ym3kj6w499	GoldenGate	REP4I	SELECT c.constraint_name, c.co...	CDB1_PDB1
0.28	42	0.01	0.50	55.06	0.00	d8y1tccn9510t	GoldenGate	EXT1A	SELECT md_tab_owner, md_tab_na...	CDB1_PDB1
0.09	42	0.00	0.16	64.99	0.00	4asccfkx961zg	GoldenGate	EXT1A	SELECT SEGCOL#, INTOCOL#, COL#...	CDBSROOT

Replication SQL ordered by User I/O Wait Time

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code for GoldenGate and XStream only
- % Total - User I/O Time as a percentage of Total User I/O Wait time
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Ordered by User I/O Time in seconds in descending order.

User I/O Time (sec)	Executions	U/O per Exec (sec)	%Total	Elapsed Time (sec)	%CPU	%IO	SQL ID	SQL Module	Process Name	SQL Text	PDB Name
0.00	2,001	0.00	0.03	0.66	92.42	0.08	38v3jc2wh3zjf	XStream	XIN4D	DELETE /*+ restrict_all_ref_c...	CDB1_PDB1
0.00	2,001	0.00	0.03	0.59	92.38	0.09	cys9jimbq3gnr	XStream	XIN3C	DELETE /*+ restrict_all_ref_c...	CDB1_PDB1
0.00	2,001	0.00	0.03	0.42	71.17	0.12	ac4shy6z3kd2b	GoldenGate	REP3H	INSERT INTO "TKGGU1"."REP3H" (...)	CDB1_PDB1
0.00	7	0.00	0.00	0.01	27.14	0.00	6ypaxx9ta5ddm	GoldenGate	REP4I	SELECT /*+ NO_STATEMENT_QUEUE...	CDB1_PDB1
0.00	21	0.00	0.00	0.05	502.04	0.00	4asccfkx961zg	GoldenGate	EXT1A	SELECT SEGCOL#, INTOCOL#, COL#...	CDB1_PDB1
0.00	7	0.00	0.00	0.03	65.39	0.00	91ckq53c2v8g5	GoldenGate	REP4I	SELECT object_name FROM all_ob...	CDB1_PDB1
0.00	7	0.00	0.00	0.01	51.84	0.00	9f8w9j7ja3um	GoldenGate	REP4I	SELECT table_name, table_type...	CDB1_PDB1
0.00	7	0.00	0.00	0.41	57.35	0.00	bn9ym3kj6w499	GoldenGate	REP4I	SELECT c.constraint_name, c.co...	CDB1_PDB1
0.00	42	0.00	0.00	0.09	64.99	0.00	4asccfkx961zg	GoldenGate	EXT1A	SELECT SEGCOL#, INTOCOL#, COL#...	CDBSROOT
0.00	2,001	0.00	0.00	0.43	83.04	0.00	bp7jxx1uwn0m	GoldenGate	REP4I	INSERT /*+ restrict_all_ref_c...	CDB1_PDB1

Replication SQL ordered by CPU Time

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code for GoldenGate and XStream only
- %Total - CPU Time as a percentage of Total DB CPU
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Ordered by CPU Time in seconds in descending order.

CPU Time (sec)	Executions	CPU per Exec (sec)	%Total	Elapsed Time (sec)	%CPU	%IO	SQL ID	SQL Module	Process Name	SQL Text	PDB Name
0.91	7	0.13	2.20	1.78	51.23	0.00	d4rmz5j1x44b	GoldenGate	REP4I	SELECT DISTINCT c.name, c.col#...	CDB1_PDB1
0.61	2,001	0.00	1.46	0.66	92.42	0.08	38v3jc2wh3zjf	XStream	XIN4D	DELETE /*+ restrict_all_ref_c...	CDB1_PDB1
0.54	2,001	0.00	1.31	0.59	92.38	0.09	cys9jimbq3gnr	XStream	XIN3C	DELETE /*+ restrict_all_ref_c...	CDB1_PDB1
0.36	2,001	0.00	0.87	0.43	83.04	0.00	bp7jxx1uwn0m	GoldenGate	REP4I	INSERT /*+ restrict_all_ref_c...	CDB1_PDB1
0.33	2,001	0.00	0.80	0.61	54.76	0.00	bnnmm4zat1nn8	GoldenGate	REP3G	INSERT INTO "TKGGU1"."REP3G" (...)	CDB1_PDB1
0.31	2,001	0.00	0.75	0.62	50.64	0.00	cdtx7d145sq4x	GoldenGate	REP3F	INSERT INTO "TKGGU1"."REP3F" (...)	CDB1_PDB1
0.30	2,001	0.00	0.72	0.42	71.17	0.12	ac4shy6z3kd2b	GoldenGate	REP3H	INSERT INTO "TKGGU1"."REP3H" (...)	CDB1_PDB1
0.25	21	0.01	0.61	0.05	502.04	0.00	4asccfkx961zg	GoldenGate	EXT1A	SELECT SEGCOL#, INTOCOL#, COL#...	CDB1_PDB1
0.23	7	0.03	0.57	0.41	57.35	0.00	bn9ym3kj6w499	GoldenGate	REP4I	SELECT c.constraint_name, c.co...	CDB1_PDB1
0.16	42	0.00	0.38	0.28	55.06	0.00	d8y1tccn9510t	GoldenGate	EXT1A	SELECT md_tab_owner, md_tab_na...	CDB1_PDB1

Replication SQL ordered by Executions

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code for GoldenGate and XStream only.
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Ordered by number of executions in descending order.

Executions	Rows Processed	Rows per Exec	Elapsed Time (sec)	%CPU	%IO	SQL Id	SQL Module	Process Name	SQL Text	PDB Name
6,003	6,003	1.00	2.33 53.1	0	0	bp7jkkz1um0m	GoldenGate	REP4I	INSERT /*+ restrict_all_ref_c...	CDB1_PDB1
240	24,000	100.00	12.15 67.7	4	4	cvt3qcbancjfp	XStream	XOUT1A	insert into sys_streams\$_apply...	CDB\$ROOT
106	676	6.38	0.83 25.7	23.8	23.8	4asccfkx961zg	GoldenGate	EXT2B	SELECT SEGCOL#, INTOCOL#, COL#...	CDB\$ROOT
96	465	4.84	6.71 41.7	1.7	1.7	4asccfkx961zg	GoldenGate	EXT2B	SELECT SEGCOL#, INTOCOL#, COL#...	CDB1_PDB1
28	28	1.00	6.93 41.7	2.6	2.6	3uuuqxkbwdycd	GoldenGate	EXT2B	declare QuerySource number := ...	CDB1_PDB1
7	7	1.00	9.59 22.9	13.5	13.5	3nb763rz02huk	GoldenGate	REP3G	SELECT status, deferrable FROM ...	CDB1_PDB1
7	7	1.00	15.43 15.9	6.1	6.1	d8w9259tpx7wh	GoldenGate	REP4J	SELECT count(*) FROM dba_const...	CDB1_PDB1
7	0	0.00	5.97 16.8	0	0	cunwhc0fb8s1z	GoldenGate	REP3G	SELECT status FROM all_constra...	CDB1_PDB1
7	0	0.00	4.80 18.8	0	0	4yvk8a1ww871h	GoldenGate	REP3G	SELECT /*+ NO_STATEMENT_QUEUEIN...	CDB1_PDB1
7	7	1.00	8.15 14	6.5	6.5	0azxj7xxq6ha2	GoldenGate	REP3H	SELECT /*+ NO_STATEMENT_QUEUEIN...	CDB1_PDB1

Oracle GoldenGate Extract Performance Metrics

This section shows the performance metrics for Extract and the capture engine.

GoldenGate Integrated Extract

- Capture name prefixed with a * indicates process (re)started between Begin and End snapshots
- Begin Lag (sec) - lag in seconds of the most recently captured message when begin snapshot was taken
- End Lag (sec) - lag in seconds of the most recently captured message when end snapshot was taken
- Redo Mined value is in number of bytes for the begin and end snapshot interval
- Ordered by Extract Name and Capture Name in ascending order

Extract Name	Capture Name	Begin Mining Lag (sec)	End Mining Lag (sec)	Redo Mined (Bytes)	Redo Mined (Bytes) per Sec	Bytes Sent per Sec	Capture LCRs	Capture LCRs per Sec	Sent LCRs	Sent LCRs per Sec	Redo Wait Times (sec)	Waiting for Extract (sec)	Waiting for Extract (delta)
EXT1A	OGG\$CAP_EXT1A	0.00	0.00	81.2M	123.4K	17.3K	50.6K	76.82	47.5K	72.21	0.00	0.00	0.00
EXT2B	OGG\$CAP_EXT2B	0.00	1.00	81.2M	123.4K	13.3K	40.2K	61.12	37K	56.14	0.00	0.00	0.00

The following table describes the performance metric data.

Note

Unless noted all data are cumulative delta collected in the snapshot interval.

Metrics Name	Description
Extract Name	GoldenGate Extract
Capture Name	Capture process name
Begin Mining Lag(s)	Lag (in seconds) derived by the time when the most recent LCR was created and received. This lag was measured at the beginning of the snapshot.
End Mining Lag(s)	Lag (in seconds) derived by the time when the most recent LCR was created and received. This lag was measured at the end of the snapshot.
Redo Mined (Bytes)	The amount of redo data mined (in bytes). If the process was restarted within the snapshot interval, then an asterisk (*) appears.
Redo Mined (Bytes) per sec	Redo Mined Rate
Bytes Sent per Sec	Number of bytes sent by the Capture process to the Extract process since the last time the Extract process attached to the Capture process.
Capture LCRs	Number of LCRs delivered to the Capture from Logminer
Capture LCRs per Sec	Capture LCRs Rate

Metrics Name	Description
Sent LCRs	Number of LCRs sent from the Capture to the Extract
Sent LCRs/Sec	Sent LCRs Rate
Redo Wait Times(sec)	Time spent by the Capture process in the WAITING FOR REDO state.
Waiting for Extract(sec)	The Capture waiting time for requests from the Extract. At the end of Snapshot
Waiting for Extract (delta)	Delta value of Waiting for Extract time between the begin snap and the end snap

Oracle GoldenGate Integrated Replicat

This section shows the performance statistics for Oracle GoldenGate integrated Replicat.

GoldenGate Integrated Replicat

- Apply name prefixed with a * indicates process (re)started between Begin and End snapshots
- LCRs processed per second by the ANR
- LCRs processed per by the reader
- Txns received by the coordinator, txns applied by the Apply server.
- Number of LCRs and LCRs applied per second
- Ordered by Replicat Name and Apply Name in ascending order

Replicat Name	Apply Name	LCRs Buffered (Reader)	Txns Received (Coordinator)	Txns Not Assigned (Coordinator)	LCRs Applied	LCRs per Sec Applied	LCRs with Dependencies %	LCRs with Watermark Dependency %	Total errors
REP4J	OGGSRREP4J	2002	1	0	2002	8.4	0	0	0

The following table describes the metric data.

Metric Name	Description
Replicat Name	Oracle GoldenGate Replicat name
Apply Name	Apply process name
LCRs Buffered (Reader)	Number of LCRs buffered waiting to be processed by Apply Reader
Txns Received (Coordinator)	Number of transactions received by Apply Coordinator
Txns Not Assigned (Coordinator)	Number of transactions not assigned yet by Apply Coordinator
LCRs Applied (Apply Server)	Number of LCRs applied by Apply Server
LCRs/Sec Applied (Apply Server)	LCRs apply rate by Apply Server
LCRs with Dependencies %	Percentage of LCRs having to wait for other transactions due to dependency
LCRs with Watermark Dependency %	Percentage of LCRs having to wait due to source transaction commit ordering
Total Errors	Total transaction with errors

Oracle GoldenGate Replicat

This section shows overall performance statistics of Oracle GoldenGate classic Replicat, coordinated Replicat, and parallel Replicat. The statistics of SQL operations are aggregated for

each Replicat process.

GoldenGate Replicat

- Includes data for Classic Replicat, Coordinated Replicat and Parallel Replicat.
- Number of rows processed by each replicat. Number of rows processed per second based on the begin and end snapshot interval.
- I/O wait time in seconds
- CPU time in seconds
- Ordered by Replicat Name in ascending order, CPU time in descending and I/O wait time in descending order.

Replicat Name	Rows Processed	Rows Processed per Sec	I/O Wait time	CPU Time
REP3F	40K	3,081.73	0.12	6.25
REP3G	222K	12,135.87	0.98	33.49
REP3H	22K	1,755.65	0.06	3.27
REP4I	114K	4,249.28	0.68	23.10

The following table describes metric data.

Metric Name	Description
Replicat Name	Replicat Name
Rows Processed	Number of rows processed by SQL operations by the Replicat
Rows Processed per Sec	Rows Processed Rate
I/O Wait Time	User I/O Wait Time of SQL operations by the Replicat
CPU Time	CPU Time of SQL operations by the Replicat

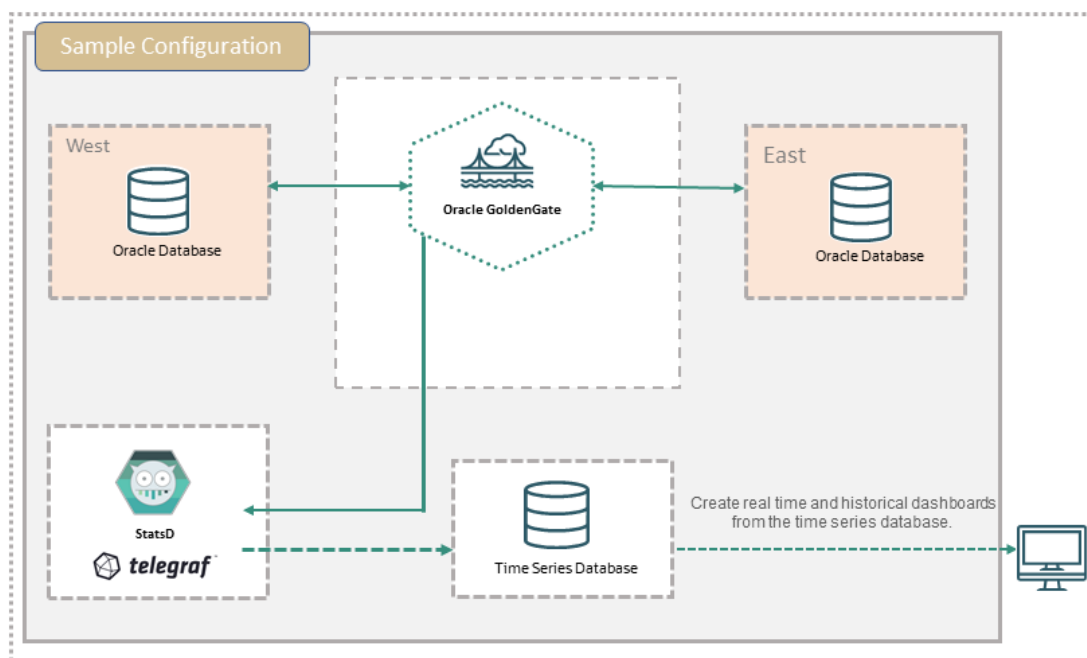
Monitor Oracle GoldenGate Statistics using StatsD

StatsD is an open-source network daemon that listens for statistics, like counters and timers, sent over TCP and sends aggregates to one or more pluggable backend services. The StatsD daemon expects metrics formatted using a specific notation.

Oracle GoldenGate Microservice Architecture sends metrics to the Telegraf service which can store data in a Time series database (TSDB).

Oracle GoldenGate generates metrics in StatsD format and transmits them directly to a network daemon that supports ingestion of StatsD-formatted metrics.

The following diagram shows the Oracle GoldenGate side and the enterprise side where the StatsD server is hosted. The performance metrics data is transformed by the Oracle GoldenGate container application and it is written to the StatsD server.

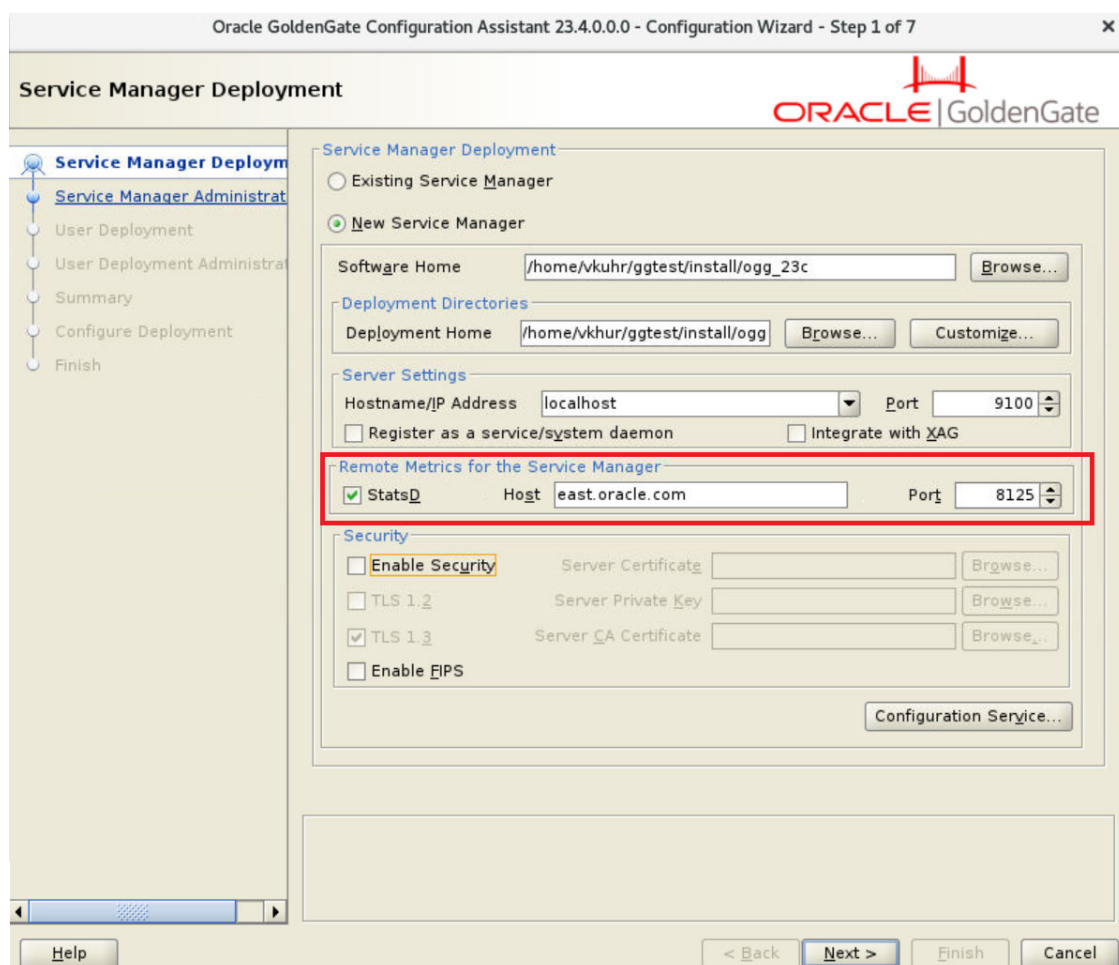


StatsD can be enabled using two ways:

- Using the OGGCA utility, you can enable StatsD for the Service Manager and deployment. See [Enable StatsD using OGGCA](#).
- Using REST API calls after the a deployment is already set up. See [Enable StatsD with REST API Service Endpoints](#)

Enable StatsD using OGGCA

StatsD can be enabled using the OGGCA utility, while configuring the Service Manager and deployments. When you run the OGGCA utility, the option to enable StatsD is available in the Service Manager Deployment screen and in the User Deployment screen, as shown in the following images:



You can use the same host for setting up StatsD for the Service Manager and deployment. The following image shows the option to enable StatsD in the User Deployment screen.

You can set up a Telegraf host on the same machine as the Oracle GoldenGate installation or you can set up a remote server to host Telegraf and receive the Oracle GoldenGate metrics from StatsD.

For details about using OGGCA, see [Add a Deployment](#).

Enable StatsD with REST API Service Endpoints

If you did not enable StatsD service while setting up the deployment and Service Manager using OGGCA, then use the following steps to enable StatsD using REST API service calls.

1. Stop the deployment:

```
curl -k -u username:password \
-d '{"status": "stopped", "enabled": false}' \
-X PATCH https://hostname:port/services/v2/deployments/deployment_name
```

2. Enable StatsD:

```
curl -svu username:password \
-X PATCH http://hostname:port/services/v2/deployments/deployment_name \
--data '{
  "metrics":
    {
      "enabled": true,
      "servers":
        [
          {
            "type": "pmsrvr",
            "protocol": "uds"
```



```

        },
        {
            "type": "statsd",
            "host": "<statsd_host>"
        }
    ]
}
}'

```

3. Start the deployment:

```

curl -k -u username:password \
-d '{"status": "running", "enabled": true}' \
-X PATCH https://hostname:port/services/v2/deployments/deployment_name

```

Also see [Update a Deployment](#).

StatsD Metrics Catalog

This section describes the metrics published by an Oracle GoldenGate deployment using the StatsD protocol. All Oracle GoldenGate StatsD metrics are reported as gauges.

Batch SQL Statistics

Name	Value
Documentation	Retrieve an existing Integrated Replicat Batch SQL Statistics
Process Type(s)	Replicat
Endpoint	/services/v2/mpoints/{item}/batchSqlStatistics
JSON Schema	mpoints:batchSqlStatistics
Base Name	batch-sql-stats
Internal Name	MPL_BATCHSQL_STATS

Tag	Description
deploymentName	The name of the OGG Microservices deployment
deploymentId	The unique identity of the OGG Microservices deployment
processType	The type of process
processName	The name of the process

Metric	Data Type	Description
total-batch-operations	integer	Total Batch Operations
total-batches	integer	Total Batches
total-batch-executions	integer	Total Batch Executions
sql-queues	integer	SQL Queues
total-batches-in-error	integer	Total Batches in Error
total-normal-mode-operations	integer	Total Normal Mode Operations
total-primary-key-collisions	integer	Total Primary Key Collisions
total-foreign-key-collisions	integer	Total Foreign Key Collisions

Metric	Data Type	Description
total-unique-key-collisions	integer	Total Unique Key Collisions
total-operation-flushes	integer	Total Operation Flushes
total-plu-flushes	integer	Total PLU Flushes
number-of-batch-thread-groups	integer	Number of Batch Thread Groups
total-commits integer Total Commits	integer	Total Commits
total-flushes	integer	Total Flushes
total-rollback	integer	Total Rollbacks
last-timestamp	integer	Last Operation Timestamp
total-elapsed-time	integer	Total Elapsed Batch Time
total-insert-time	integer	Total Insert Time
total-update-time	integer	Total Update Time
total-delete-time	integer	Total Delete Time

Bounded Recovery Status

Name	Value
Documentation	Retrieve an existing Bounded Recovery Status
Process Type(s)	Extract
Endpoint	/services/v2/mpoints/{item}/brStatus
JSON Schema	mpoints:brStatus
Base Name	br-status
Internal Name	MPL_BR_STATUS

Tag	Description
deploymentName	The name of the OGG Microservices deployment
deploymentId	The unique identity of the OGG Microservices deployment
processType	The type of process
processName	The name of the process
br-current-status	The status of the BR
force-checkpoint-time	Force Checkpoint time

Metric	Data Type	Description
last-checkpoint-number	integer	Last checkpoint time
checkpoint-intervalseconds	integer	The time interval between two checkpoints
total-num-objectspersisted	integer	Number of objects to be persisted at the BR checkpoint
total-state-bytespersisted	integer	State size of all objects to be persisted at the BR checkpoint

Metric	Data Type	Description
total-data-bytespersisted	integer	Data size of all objects to be persisted at the BR checkpoint
outstanding-numobjects	integer	Number of objects still remaining to be persisted at the BR checkpoint
outstanding-statebytes	integer	State size of all objects still remaining to be persisted at the BR checkpoint
outstanding-databytes	integer	Data size of all objects still remaining to be persisted at the BR checkpoint

Cache Manager Statistics

Name	Value
Documentation	Retrieve an existing Cache Manager Statistics
Process Type(s)	Extract, Replicat
Endpoint	/services/v2/mpoints/{item}/cacheStatistics
JSON Schema	mpoints:cacheStatistics
Base Name	cache-statistics
Internal Name	MPL_CACHE_STATS

Tag	Description
deploymentName	The name of the OGG Microservices deployment
deploymentId	The unique identity of the OGG Microservices deployment
processType	The type of process
processName	The name of the process

Metric	Data Type	Description
total-objects-in-cache	integer	Objects in Cache
total-objects	integer	Total Cached Objects
total-objects-active	integer	Total Active Objects
total-objects-committed	integer	Total Committed Objects
max-active-objects	integer	Maximum number of Active Objects
times-buffer-overflowed	integer	Times Cache Buffer Overflowed
times-get-next-from-file	integer	Number of Get Next From File Operations
times-get-last-from-file	integer	Number of Get Last From File Operations
times-small-buff-forced-out	integer	Number of Times a Small Buffer was Forced Out
times-retrieved	integer	Number of Retrieval Operations
total-number-of-q-hits	integer	Number of Cache Hits
total-number-of-q-misses	integer	Number of Cache Misses
total-number-of-q-puts	integer	Number of Cache Puts

Metric	Data Type	Description
total-number-of-q-tries	integer	Number of Cache Tries
total-number-of-q-entries	integer	Number of Queue Tries
max-number-of-q-entries	integer	Maximum number of Queue Tries
total-munmap	integer	Number of UnMap Operations
total-cnnbl-attempts	integer	Number of Cannibalize Attempts
total-cnnbl-success	integer	Number of Successful Cannibalize Attempts
total-cnnbl-mbufs	integer	Number of Buffer Cannibalize Attempts
total-file-cache-requests	integer	Number of File Cache Requests
total-file-cache-entries	integer	Number of File Cache Entries
total-file-cache-placed	integer	Number of File Objects Placed in Cache
max-qlength	integer	Maximum Queue Length
max-processed	integer	Maximum number of operations processed
times-wait-signaled	integer	Number of times a Cache Operation was Wait Signaled
times-file-cache-not-needed	integer	Number of times a File Cache was not needed
times-requestor-needed-fc	integer	Number of times a Requestor Needed a File Cache
total-objects-in-file-cache	integer	Total Number of Objects in the File Cache
total-file-cache-bytes-todisk	integer	Size of the File Cache on Disk
times-cache-flushed	integer	Number of times the Cache was flushed
max-memory-usage	integer	Maximum Memory Usage for the Cache
average-memory-usage	integer	Average Memory Usage for the Cache

Distribution Service Network Statistics

Name	Value
Documentation	Retrieve an existing Distribution Service Network Statistics
Process Type(s)	Distribution Service
Endpoint	/services/v2/mpoints/{item}/distsrvrNetworkStats
JSON Schema	mpoints:distsrvrNetworkStats
Base Name	distributionServer-network-stats
Internal Name	MPL_DS_NET_STATS

Tag	Description
deploymentName	The name of the OGG Microservices deployment
deploymentId	The unique identity of the OGG Microservices deployment
processType	The type of process
processName	The name of the process
path-name	The name of the path

total-receive-time	integer	Receive Wait Time
total-send-time	integer	Send Wait Time
total-bytes-sent	integer	Total Number of Bytes Sent
total-bytes-received	integer	Total Number of Bytes Received
total-messages-received	integer	Total Number of Messages Received
total-messages-sent	integer	Total Number of Messages Sent

Distribution Service Path Statistics

Name	Value
Documentation	Retrieve an existing Distribution Service Path Statistics
Process Type(s)	Distribution Service
Endpoint	/services/v2/mpoints/{item}/distsrvrPathStats
JSON Schema	mpoints:distsrvrPathStats
Base Name	distributionServer-path-stats
Internal Name	MPL_DS_PATH_STATS

Tag	Description
deploymentName	The name of the OGG Microservices deployment
deploymentId	The unique identity of the OGG Microservices deployment
processType	The type of process
processName	The name of the process
path-name	The name of the path

Metric	Data Type	Description
lcrs-received	integer	Number of LCRs Received
lcrs-sent	integer	Number of LCRs Sent
total-ddls	integer	Total Number of DDLs
total-procedures	integer	Total Number of Procedures
total-inserts	integer	Total Number of Inserts
total-updates	integer	Total Number of Updates
total-upserts	integer	Total Number of Upserts
total-deletes	integer	Total Number of Deletes
total-unsupported	integer	Total Number of Unsupported
total-others	integer	Total Number of Other Commands

Distribution Service Table Statistics

Name	Value
Documentation	Retrieve an existing Distribution Service Table Statistics

Name	Value
Process Type(s)	Distribution Service
Endpoint	/services/v2/mpoints/{item}/distsrvrTableStats
JSON Schema	mpoints:distsrvrTableStats
Base Name	distributionServer-table-stats
Internal Name	MPL_DS_TABLE_STATS

Tag	Description
deploymentName	The name of the OGG Microservices deployment
deploymentId	The unique identity of the OGG Microservices deployment
processType	The type of process
processName	The name of the process
path-name	The name of the path
table-name	N/A

Metric	Data Type	Description
lcrs-received	integer	N/A
lcrs-sent	integer	N/A
total-inserts	integer	N/A
total-updates	integer	N/A
total-upserts	integer	N/A
total-truncates	integer	N/A
total-deletes	integer	N/A
total-unsupported	integer	N/A
total-others	integer	N/A

Checkpoint Position Information Statistics

Name	Value
Documentation	Retrieve an existing Checkpoint Position Information
Process Type(s)	Extract, Replicat
Endpoint	/services/v2/mpoints/{item}/positionEr
JSON Schema	mpoints:positionEr
Base Name	position-er
Internal Name	MPL_ER_POSITION

Tag	Description
deploymentName	The name of the OGG Microservices deployment
deploymentId	The unique identity of the OGG Microservices deployment
processType	The type of process

Tag	Description	
processName	The name of the process	

Metric	Data Type	Description
low-watermark-lag	integer	Lag of the low watermark (milliseconds)
high-watermark-lag	integer	Lag of the high watermark (milliseconds)
last-operation-lag	integer	Lag of last operation (milliseconds)
end-of-file	boolean	End of file indicator
trail-times-at-eof	integer	Number of Times Read Hit End of File

Input Trail File Statistics

Name	Value
Documentation	Retrieve an existing Input Trail File Statistics
Process Type(s)	Replicat, Distribution Service
Endpoint	/services/v2/mpoints/{item}/trailInput
JSON Schema	mpoints:trailInput
Base Name	trail-input
Internal Name	MPL_ER_TRAIL_IN

Tag	Description
deploymentName	The name of the OGG Microservices deployment
deploymentId	The unique identity of the OGG Microservices deployment
processType	The type of process
processName	The name of the process
trail-name	The base name of the trail
trail-path	The path for storing the trail

Metric	Data Type	Description
io-read-count	integer	Number of I/O Read Operations
io-read-bytes	integer	Number of I/O Read Bytes Transferred
trail-read-errors	integer	Number of I/O Read Errors
trail-times-at-eof	integer	Number of Times Read Hit End of File
trail-lob-bytes	integer	Number of LOB bytes read in trail file
trail-read-time	integer	Time spent in read of trail file
trail-rba	integer	Current RBA
trail-seq	integer	Current Sequence

Output Trail File Statistics

Name	Value
Documentation	Retrieve an existing Output Trail File Statistics
Process Type(s)	Extract
Endpoint	/services/v2/mpoints/{item}/trailOutput
JSON Schema	mpoints:trailOutput
Base Name	trail-output
Internal Name	MPL_ER_TRAIL_OUT

Tag	Description
deploymentName	The name of the OGG Microservices deployment
deploymentId	The unique identity of the OGG Microservices deployment
processType	The type of process
processName	The name of the process
trail-name	The base name of the trail
trail-path	The path for storing the trail

Metric	Data Type	Description
port-number	integer	The TCP/IP port for the remote trail
port-type	integer	The type of port connection
io-write-count	integer	Number of I/O Write Operations
io-write-bytes	integer	Number of I/O Write Bytes Transferred
trail-write-time	integer	Time spent in write of trail file
trail-lob-bytes	integer	Number of LOB bytes write in trail file
trail-buffer-flushes	integer	Number of trail file buffer flushes
trail-buffer-flush-time	integer	Time spent flushing trail file buffers
trail-bytes-flushed	integer	Number of bytes flushed from trail file buffers
trail-rba	integer	Current RBA
trail-seq	integer	Current Sequence
trail-max-bytes	integer	Maximum number of bytes for a Trail file

Extract Database Statistics

Name	Value
Documentation	Retrieve an existing Extract Database Statistics
Process Type(s)	Extract
Endpoint	/services/v2/mpoints/{item}/statisticsExtract
JSON Schema	mpoints:statisticsExtract
Base Name	statistics-extract

Name	Value
Internal Name	MPL_EX_COUNTS

Tag	Description
deploymentName	The name of the OGG Microservices deployment
deploymentId	The unique identity of the OGG Microservices deployment
processType	The type of process
processName	The name of the process

Metric	Data Type	Description
mapped-total-inserts	integer	Number of INSERT operations performed
mapped-total-updates	integer	Number of UPDATE operations performed
mapped-total-deletes	integer	Number of DELETE operations performed
mapped-total-upserts	integer	Number of UPSERT operations performed
mapped-total-unsupported	integer	Number of UNSUPPORTED operations performed
mapped-total-truncates	integer	Number of TRUNCATE operations performed
total-executed-ddls	integer	Number of DDL operations performed
total-executed-procedures	integer	Number of Procedures operations performed
total-discards	integer	Number of operations discarded
total-ignores	integer	Number of errors ignored
total-conversion-errors	integer	Number of character conversion errors
total-conversion-truncates	integer	Number of character conversion truncations
total-row-fetch-attempts	integer	Number of attempts to fetch from DBMS
total-row-fetch-failures	integer	Number of failed fetch attempts
total-metadata-records	integer	Number of Metadata records written to trail

Network Statistics

Name	Value
Documentation	Retrieve an existing Network Statistics
Process Type(s)	Extract
Endpoint	/services/v2/mpoints/{item}/networkStatistics
JSON Schema	mpoints:networkStatistics
Base Name	network-stats
Internal Name	MPL_NET_STATS

Tag	Description
deploymentName	The name of the OGG Microservices deployment
deploymentId	The unique identity of the OGG Microservices deployment
processType	The type of process

Tag	Description	
processName	The name of the process	
host-name	The name of the host	

Metric	Data Type	Description
port-number	integer	Port Number
socket-open-time	integer	N/A
inbound-bytes	integer	Inbound Bytes Received
inbound-messages	integer	Inbound Messages Received
outbound-bytes	integer	Outbound Bytes Sent
outbound-messages	integer	Outbound Messages Sent
send-wait-time	integer	Send Wait Time
receive-wait-time	integer	Receive Wait Time
send-count	integer	Send Count
receive-count	integer	Receive Count
compression-wc-time	integer	Compression Wallclock Time
compression-cpu-time	integer	Compression CPU Time
decompression-wc-time	integer	Decompression Wallclock Time
decompression-cpu-time	integer	Decompression CPU Time
encryption-wc-time	integer	Encryption Wallclock Time
encryption-cpu-time	integer	Encryption CPU Time
encryption-bytes	integer	Encryption Bytes
decryption-wc-time	integer	Decryption Wallclock Time
decryption-cpu-time	integer	Decryption CPU Time
decryption-bytes	integer	Decryption Bytes
compresses_bytes_in	integer	Compress Bytes In
uncompresses_bytes_in	integer	Uncompress Bytes In
compresses_bytes_out	integer	Compress Bytes Out
uncompresses_bytes_out	integer	Uncompress Bytes Out

Process Performance Statistics

Name	Value
Documentation	Retrieve an existing Process Performance Resource Utilization Information
Process Type(s)	All
Endpoint	/services/v2/mpoints/{item}/processPerformance
JSON Schema	mpoints:processPerformance
Base Name	process-performance
Internal Name	MPL_PROC_PERF

Tag	Description	
deploymentName	The name of the OGG Microservices deployment	
deploymentId	The unique identity of the OGG Microservices deployment	
processType	The type of process	
processName	The name of the process	

Metric	Data Type	Description
process-start-time	integer	Process Start Time
process-id	integer	The operating system process id
thread-count	integer	Number of Threads
handle-count	integer	Number of Handles
cpu-time	integer	Process CPU Time (in microseconds)
kernel-time	integer	Process Kernel Mode Time (in microseconds)
user-time	integer	Process User Mode Time (in microseconds)
io-read-count	integer	Number of I/O Read Operations
io-write-count	integer	Number of I/O Write Operations
io-other-count	integer	Number of I/O Other Operations
io-read-bytes	integer	Number of I/O Read Bytes Transferred
io-write-bytes	integer	Number of I/O Write Bytes Transferred
io-other-bytes	integer	Number of I/O Other Bytes Transferred
page-faults	integer	Number of Page Faults for the Process
working-set-size	integer	Process Working Set Size in bytes
peak-working-set-size	integer	Peak Process Working Set Size in bytes
private-bytes	integer	Process Private Size in bytes

Queue Statistics

Name	Value
Documentation	Retrieve an existing Queue Statistics
Process Type(s)	Extract
Endpoint	/services/v2/mpoints/{item}/queueStatistics
JSON Schema	mpoints:queueStatistics
Base Name	queue-stats
Internal Name	MPL_QUEUE_STATS

Tag	Description
deploymentName	The name of the OGG Microservices deployment
deploymentId	The unique identity of the OGG Microservices deployment
processType	The type of process
processName	The name of the process

Tag	Description	
queue-name	The name of the queue	

Metric	Data Type	Description
queue-items	integer	Items in the Queue
queue-reads	integer	Queue Reads
queue-read-waits	integer	Queue Reads resulting in a wait
queue-reads-signaled	integer	Queue Read Waits that were signaled
queue-writes	integer	Queue Writes
queue-write-waits	integer	Queue Writes resulting in a wait
queue-writes-signaled	integer	Queue Write Waits that were signaled
queue-max-size	integer	Maximum size of the Queue
queue-current-size	integer	Current size of the Queue
thread-id	integer	Owning Thread id

Replicat Database Statistics

Name	Value
Documentation	Retrieve an existing Replicat Database Statistics
Process Type(s)	Replicat
Endpoint	/services/v2/mpoints/{item}/statisticsReplicat
JSON Schema	mpoints:statisticsReplicat
Base Name	statistics-replicat
Internal Name	MPL_RP_COUNTS

Tag	Description
deploymentName	The name of the OGG Microservices deployment
deploymentId	The unique identity of the OGG Microservices deployment
processType	The type of process
processName	The name of the process

Metric	Data Type	Description
mapped-total-inserts	integer	Number of INSERT operations performed
mapped-total-updates	integer	Number of UPDATE operations performed
mapped-total-deletes	integer	Number of DELETE operations performed
mapped-total-upserts	integer	Number of UPSERT operations performed
mapped-totalunsupported	integer	Number of UNSUPPORTED operations performed
mapped-totaltruncates	integer	Number of TRUNCATE operations performed
total-executed-ddls	integer	Number of DDL operations performed
total-executedprocedures	integer	Number of Procedures operations performed

Metric	Data Type	Description
total-discards	integer	Number of operations discarded
total-ignores	integer	Number of errors ignored
total-conversion-errors	integer	Number of character conversion errors
total-conversiontruncates	integer	Number of character conversion truncations
total-conflictsdetected	integer	Number of database conflicts
total-conflicts-resolved	integer	Number of database conflicts resolved per user settings
total-conflicts-failed	integer	Number of database conflicts that could not be automatically resolved

Parallel Replicat Statistics

Name	Value
Documentation	Retrieve an existing Parallel Replicat Statistics
Process Type(s)	Replicat
Endpoint	/services/v2/mpoints/{item}/parallelReplicat
JSON Schema	mpoints:parallelReplicat
Base Name	parallel-replicat-stats
Internal Name	MPL_RP_PARALLEL

Tag	Description
deploymentName	The name of the OGG Microservices deployment
deploymentId	The unique identity of the OGG Microservices deployment
processType	The type of process
processName	The name of the process

Metric	Data Type	Description
total-transactions	integer	Total Transactions Processed
split-transactions	integer	Split Transactions
split-avg-chunks	integer	Split transactions average splits
split-avg-rows	integer	Split transactions average rows
serialize-transactions	integer	Transactions serialized
large-transactions	integer	Total Number of Large Transactions
large-avg-rows	integer	Large Transactions average rows
large-avg-chunks	integer	Large Transactions average number of chunks
missing-dep-cols	integer	Missing dependency columns
non-native-support	integer	Transactions with non-native support
max-scheduling-const	integer	Transactions with number of constraints greater than the maximum supported
metadata-changes	integer	Number of Metadata Barriers

Metric	Data Type	Description
ddl-operation-barriers	integer	Number of DDL Fallbacks
sequence-operation-barriers	integer	Number of Sequence Fallbacks
proc-ops	integer	N/A
event-actions	integer	Number of Event Fallbacks
stored-procedures	integer	Number of Stored Procedure Fallbacks
synchronous	integer	Transactions serialized due to SYNCHRONOUS option
fk-parent-row-deletes	integer	Dependencies due to parent row delete operations
table-level-dependencies	integer	Table-level dependencies
virtual-column-dependencies	integer	Dependencies due to virtual columns
fallbacks-transactions	integer	Total Transactions Processed
active-appliers	integer	Number of active Appliers

Receiver Service Statistics

Name	Value
Documentation	Retrieve an existing Receiver Service Statistics
Process Type(s)	Receiver Service
Endpoint	/services/v2/mpoints/{item}/recvsrvrStats
JSON Schema	mpoints:recvsrvrStats
Base Name	receiver-service-stats
Internal Name	MPL_RS_STATS

Tag	Description
deploymentName	The name of the OGG Microservices deployment
deploymentId	The unique identity of the OGG Microservices deployment
processType	The type of process
processName	The name of the process
path-name	The name of the path
target-file-name	Target File Name
last-file	Last File Received

Metric	Data Type	Description
total-bytes-written	integer	Total File Bytes Written
total-commands	integer	Total Number of Commands
total-sends	integer	Total Number of Sends
total-receives	integer	Total Number of Receives
total-bytes-sent	integer	Total Number of Bytes Sent
total-bytes-received	integer	Total Number of Bytes Received

Metric	Data Type	Description
total-messages-sent	integer	Total Number of Messages Sent
total-messages-received	integer	Total Number of Messages Received
total-send-wait-time	integer	Total Send Wait Time
total-receive-wait-time	integer	Total Receive Wait Time
send-buffer-size	integer	Size of the Send Buffer
receive-buffer-size	integer	Size of the Receive Buffer
message-buffer-size	integer	Size of the Message Buffer
flush-size-threshold	integer	Threshold size for flushing the Buffer
flush-time-threshold	integer	Threshold time for flushing the Buffer
is-alive	boolean	The path is connected to the Distribution Server

Extract Database Statistics by Table

Name	Value
Documentation	Retrieve an existing Extract Database Statistics by Table
Process Type(s)	Extract
Endpoint	/services/v2/mpoints/{item}/statisticsTableExtract
JSON Schema	mpoints:statisticsTableExtract
Base Name	statistics-table-extract
Internal Name	MPL_TABLE_COUNTS_EX

Tag	Description
deploymentName	The name of the OGG Microservices deployment
deploymentId	The unique identity of the OGG Microservices deployment
processType	The type of process
processName	The name of the process
table-name	The name of the table

Metric	Data Type	Description
mapped-total-inserts	integer	Number of INSERT operations performed
mapped-total-updates	integer	Number of UPDATE operations performed
mapped-total-deletes	integer	Number of DELETE operations performed
mapped-total-truncates	integer	Number of TRUNCATE operations performed
total-ignores	integer	Number of errors ignored
total-discards	integer	Number of operations discarded
total-row-fetch-attempts	integer	Number of attempts to fetch from DBMS
total-row-fetch-failures	integer	Number of failed fetch attempts

Replicat Database Statistics by Table

Name	Value
Documentation	Retrieve an existing Replicat Database Statistics by Table
Process Type(s)	Replicat
Endpoint	/services/v2/mpoints/{item}/statisticsTableReplicat
JSON Schema	mpoints:statisticsTableReplicat
Base Name	statistics-table-replicat
Internal Name	MPL_TABLE_COUNTS_RP

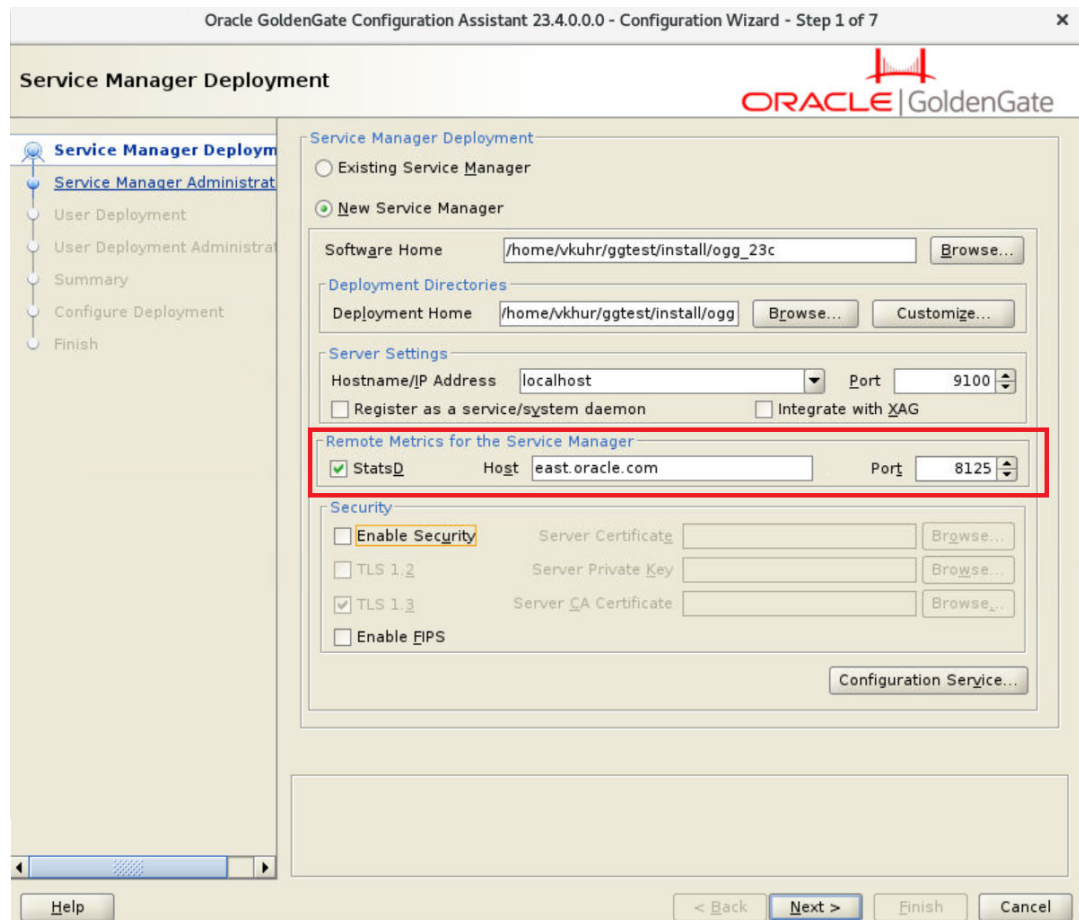
Tag	Description
deploymentName	The name of the OGG Microservices deployment
deploymentId	The unique identity of the OGG Microservices deployment
processType	The type of process
processName	The name of the process
table-name	The name of the table
target-table-name	The name of the target table

Metric	Data Type	Description
mapped-total-inserts	integer	Number of INSERT operations performed
mapped-total-updates	integer	Number of UPDATE operations performed
mapped-total-deletes	integer	Number of DELETE operations performed
mapped-total-upserts	integer	Number of UPSERT operations performed
mapped-total-truncates	integer	Number of TRUNCATE operations performed
total-ignores	integer	Number of errors ignored
total-discards	integer	Number of operations discarded
total-conflicts-detected	integer	Total Apply Conflicts
total-conflicts-resolved	integer	Total Conflicts Automatically Resolved
total-conflicts-failed	integer	Total Conflicts that Failed Automatic Resolution
total-insert-row-exists	integer	Number of database conflicts of the type insert row exists
total-update-row-exists	integer	Number of database conflicts of the type update row exists
total-update-row-missing	integer	Number of database conflicts of the type update row missing
total-delete-row-exists	integer	Number of database conflicts of the type delete row exists
total-delete-row-missing	integer	Number of database conflicts of the type delete row missing

Disable StatsD using OGGCA

To disable the StatsD service for an existing deployment, run the OGGCA configuration wizard and perform the following steps:

1. Deselect the StatsD option from the Service Manager Deployment screen.



2. Click Next and finish the steps in OGGCA wizard.

For details about using OGGCA, see [Add a Deployment](#).

Disable StatsD with REST API Service Endpoints

To disable the StatsD service in an existing deployment using REST API service calls, perform the following steps:

1. Stop the deployment:

```
curl -k -u username:password \
-d '{"status": "stopped", "enabled": true}' \
-X PATCH https://hostname:port/services/v2/deployments/deployment_name
```

2. Disable StatsD:

```
curl -svu username:password \
-X PATCH http://hostname:port/services/v2/deployments/deployment_name \
--data '{
    "metrics":
        {
            "enabled":false,
            "servers":
                [
                    {
                        "type":"pmsrvr",
                        "protocol":"uds"
                    },
                    {
                        "type":"statsd",
                        "protocol":"uds"
                    }
                ]
        }
    "host":"statsd_host"
}'
```

3. Start the deployment:

```
curl -k -u username:password \
-d '{"status": "running","enabled": true}' \
-X PATCH https://hostname:port/services/v2/deployments/deployment_name
```

Also see [Update a Deployment](#).

About Integrated Diagnostics

The Integrated Diagnostics REST API helps obtain the performance data to be able to diagnose performance issues related to integrated Extract, integrated Replicat, and parallel Replicat integrated Replicat. This is a replacement over the previously used Replication Performance Advisory Utility (UTLRPDV).

Integrated Diagnostics feature is responsible for collecting diagnostic and performance data for each of the Oracle GoldenGate Extract and Replicat processes. This process preserves interval data in memory and computes all statistical values related to CPU usage, session wait times, various counts and status.

The usage of the Integrated Diagnostics feature has two parts:

- Collecting data
- Displaying data

There can be multiple instances of Integrated Diagnostics running, which collect data for a different replication process. The Diagnostic Collector will write the collected diagnostic data in JSON format to the {deployment}/var/lib/reports directory. The filename is in the format: {ProcessName}.diagnostics.YYYY-MM-DD-HH-MM-SSZ

A maximum of 10 collection files for each process will exist. When a new collection starts, if there are already 10 files, the least current collection will be purged. Collected diagnostics can

be accessed while the collector is running as it saves the data after every sample is collected. You can download the existing reports at any time.

The Diagnostic Collector will only allow one instance to be running for a specific Extract or Replicat at a time. The Administrative Service retrieves the collected diagnostic and performance data and provide listings of previously collected data.

How to Configure Integrated Diagnostics

To set up Extract or Replicat monitoring with the Integrated Diagnostics feature, perform the following steps:

1. From the Administration Service, left-navigation pane, select the Extract or Replicat process for which you want to configure performance monitoring with Integrated Diagnostics. The process menu expands in the left pane.
2. Click the **Integrated Diagnostics** option to open the configuration page for Integrated Diagnostics.
3. Click the plus (+) sign to open the Create Integrated Diagnostics Collection dialog box.

Create Integrated Diagnostics Collection



Create an integrated diagnostics collection task for the specific integrated extract or replicat. This will gather performance related statistics of the capture or apply engine in the database.

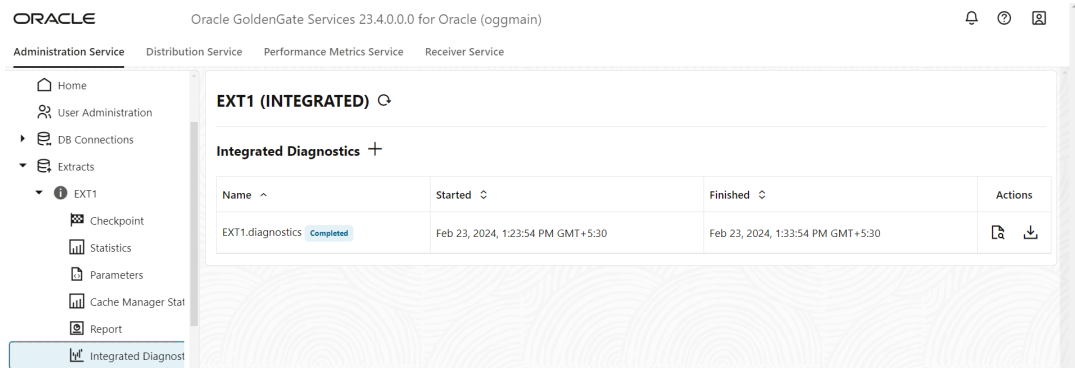
Collection Frequency (secs) 5	Collection Duration (mins) 10

Submit

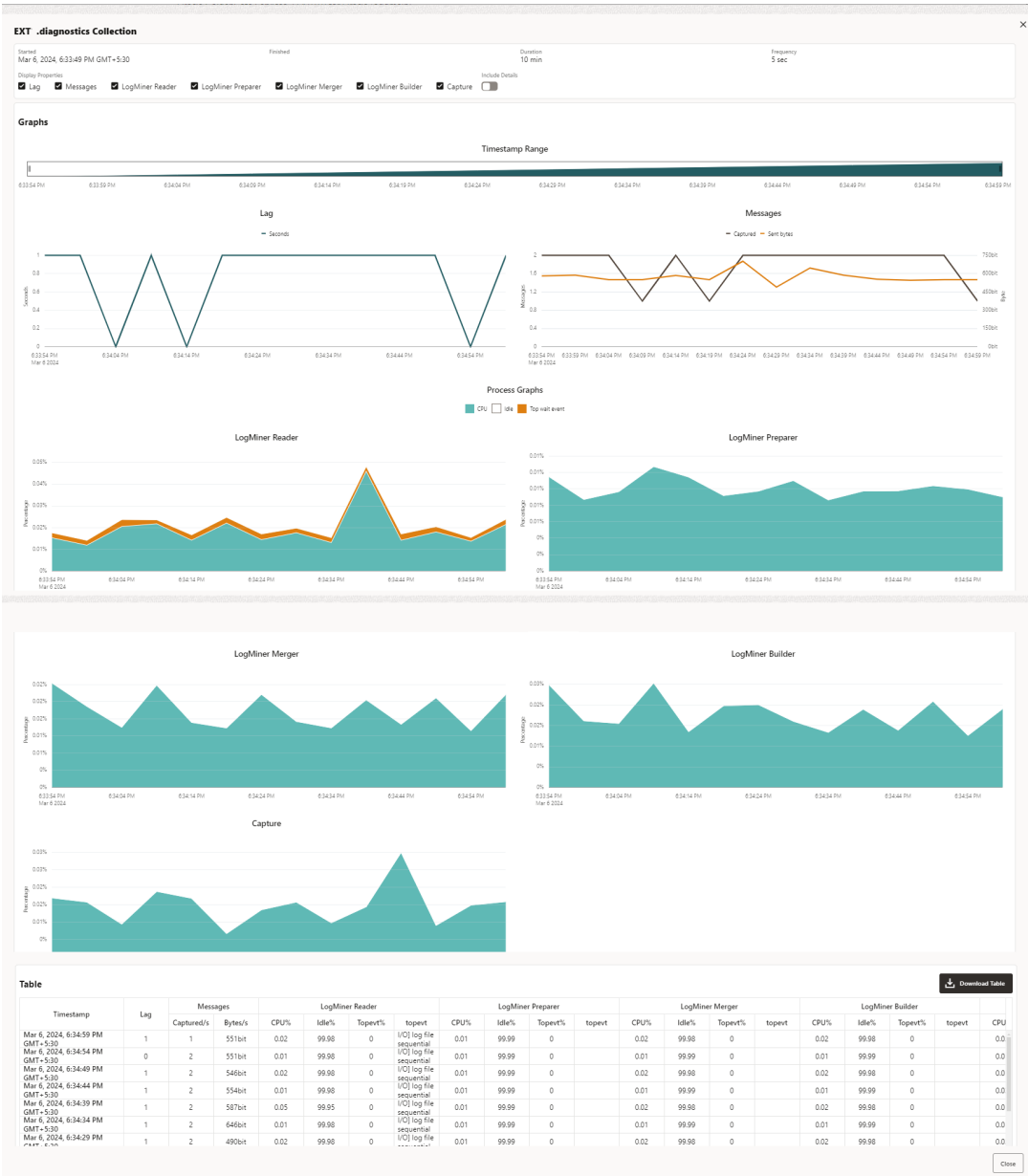
4. Specify the following values in this dialog box:
 - **Collection Frequency** (in seconds): Sets the frequency time interval for collecting performance statistics for the selected Extract/Replicat process. Minimum value is 5 seconds.
 - **Collection Duration** (in minutes): Set the duration for which the data is collected.

For example, if you collect data in a 10 second interval for 5 minutes (300 seconds), the collection contains 30 measurement points.

- Click **Submit** to create a query. This query will run for the specified Collection Duration. Notice that the query shows in Running state till the collection duration is over. The status changes to Completed after the data collection duration is over, as shown in the following image.



- Click the **Details** icon to view the detailed results of the query. A dialog box is displayed that shows the monitoring data for the selected process.



As shown in the image, the information is depicted using graphical charts and a table as well. You can download this table as a .csv file also.

The information displayed includes measuring(Display Properties). idle time, top-wait event, CPU usage, System I/O file sequential read values for Extract, Replicat processes. These values are measured for Lag, messages Captured/sent, LogMiner reader, LogMiner Preparer, LogMiner Merger, LogMiner Builder, and the Capture or Replicat process.

7. Use the **Download** icon, to download the results of the query.

Monitor Processes and Statistics from the Command Line

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 Microservices](#).

To learn about command syntax, usage, and examples, see the *Command Line Interface Reference* for Oracle GoldenGate.

Command	What it Shows
INFO {EXTRACT REPLICAT} <i>group</i> [DETAIL]	Run status, checkpoints, approximate lag, and environmental information.
INFO ALL	Displays the INFO output for all Oracle GoldenGate processes on the system.
STATS {EXTRACT REPLICAT} <i>group</i>	Displays statistics on processing volume, such as number of operations performed.
STATUS {EXTRACT REPLICAT} <i>group</i>	Displays the run status (starting, running, stopped, abended) for Extract and Replicat processes.
LAG {EXTRACT REPLICAT} <i>group</i>	Displays the latency between last record processed and timestamp in the data source.
INFO {EXTTRAIL} <i>trail</i>	Displays the name of associated process, position of last data processed, maximum file size.
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.
SEND {EXTRACT REPLICAT } <i>group</i>	Depending on the process and selected options, returns information about memory pool, lag, TCP statistics, long-running transactions, process status, recovery progress, and more.
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>	Information dependent on the COMMAND type: INFO LAG SEND STATS STATUS <i>wildcard</i> is a wildcard specification for the process groups to be affected, for example: INFO ER ext* STATS ER *
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.

Command	What it Shows
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.
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 DBLOGIN 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.
STATS DISTPATH RECVPATH	Get the statistics for the distribution path (DISTPATH) or receiver path (RECVPATH).
STATS ER	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.
STATUS ER	Checks the status of multiple Extract and Replicat groups as a unit.
STATUS DEPLOYMENT	View the status of the specified deployment.
STATUS PMSRVR	Status of Performance Service.
STATUS SERVICE	Displays the status of specified Oracle GoldenGate service.

Monitor Oracle GoldenGate Processes, Trails, and Paths

Learn about how to monitor Oracle GoldenGate processes, trails, and paths.

Search and Read the Log Information from the Diagnosis Page

Log information allows you to monitor all the messages logged for your Service Manager. This includes processes, trails, paths, microservices, and deployments managed from the Service Manager.

Collective log information for all processes, trails, and paths associated with all deployments and microservices can be accessed from the **Diagnosis** page in Service Manager. Log information includes details such as the following:

- Lag information for Extract, Replicat processes, which provides the latency value between the last record processed and its timestamp in the data source
- Heartbeat table activities from the heartbeat history table. Also see [Monitor Lag Using Automatic Heartbeat Tables](#)
- Status messages for Oracle GoldenGate processes, trails, and paths
- Error messages for Oracle GoldenGate processes, trails, and paths
- Status of deployments and microservices
- Error messages of deployments or microservices
- Heartbeat

You can perform the following tasks on this page:

- Sort the Log Information table by column
- Refresh the log using the Refresh button
- Search for specific log messages using the search criteria as date, severity, and message

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

Access the **Diagnosis** page from the left navigation pane of the Service Manager. The complete log information is displayed on the page.

Search for Log Messages

If you want need to search for a specific message, you can also search for it by following these steps:

1. Enter a search criteria in the **Search** box. The search criteria can be **Date**, or **Severity** of the message(s), or the Message string itself. You can add multiple search criteria in the search box.
2. If you select **Date**, then you get the options:
 - Starting on: Displays the log information from the specified start date.
 - Ending on: Displays the log information till the specified end date.
 - Range between: Displays the log information between the start and end date range.

Service Log (phoenix433404)

Select Service
Administration Service

Search By:

☐ Starting on ☐ Ending on ☐ Range Between

Date	Severity	Message
Feb 13 2024 11:37:26	1	1 Ordered trustpoints. (Thread 8)
Feb 13 2024 11:37:25	1	1 Ordered trustpoints. (Thread 9)
Feb 13 2024 11:37:25	1	1 Ordered trustpoints. (Thread 10)
Feb 13 2024 11:37:23	1	1 Ordered trustpoints. (Thread 6)
Feb 13 2024 11:37:22	1	1 Ordered trustpoints. (Thread 7)
Feb 13 2024 11:37:22	1	1 Ordered trustpoints. (Thread 8)
Feb 13 2024 11:37:22	1	1 Ordered trustpoints. (Thread 9)
Feb 13 2024 11:37:17	1	1 Ordered trustpoints. (Thread 6)
Feb 13 2024 11:37:17	1	1 Ordered trustpoints. (Thread 8)

The screen shows the **Date** search criteria with the **Starting on** date.

- If you select **Severity** of the message in the log, then you get to choose from the following levels of severity:

ORACLE Oracle GoldenGate 26.1.0.0.0 for Oracle (ServiceManager)

Service Log (phoenix433404)

Select Service
Administration Service

Search By:

Severity

Date	Severity	Message
Feb 13	WARN AND UP	1 Ordered trustpoints. (Thread 10)
Feb 13	FATAL	1 Ordered trustpoints. (Thread 9)
Feb 13	REPORT	1 Ordered trustpoints. (Thread 6)
Feb 13	INFO	1 Ordered trustpoints. (Thread 7)
Feb 13	DEBUG	1 Ordered trustpoints. (Thread 10)
Feb 13	SUCCESS	1 Ordered trustpoints. (Thread 6)
Feb 13	ERROR	1 Ordered trustpoints. (Thread 9)
Feb 13	ERROR	1 Ordered trustpoints. (Thread 8)
Feb 13 2024 13:55:52		1 Ordered trustpoints. (Thread 7)

The screen shows various severity levels of messages. You can select any of these severity levels and search for log messages.

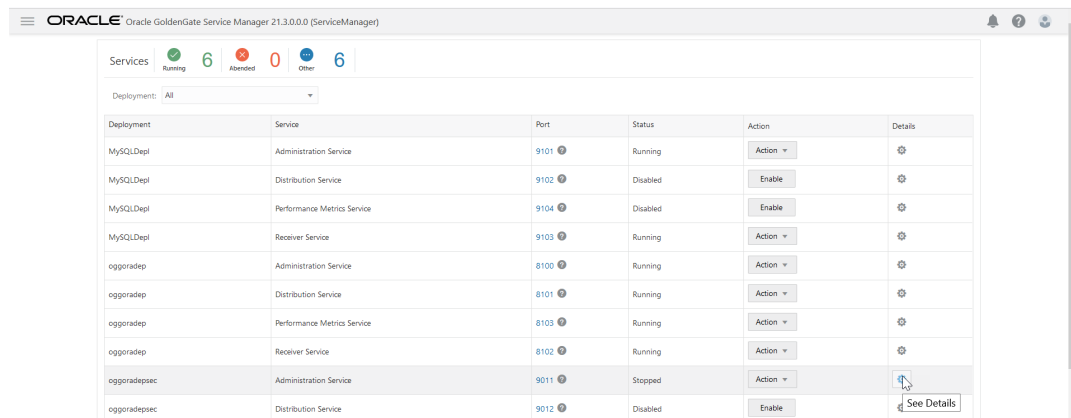
- Warn
- Fatal
- Report
- Info
- Debug
- Success
- Error

- Click **Clear Filter** if you want to delete the search criteria.

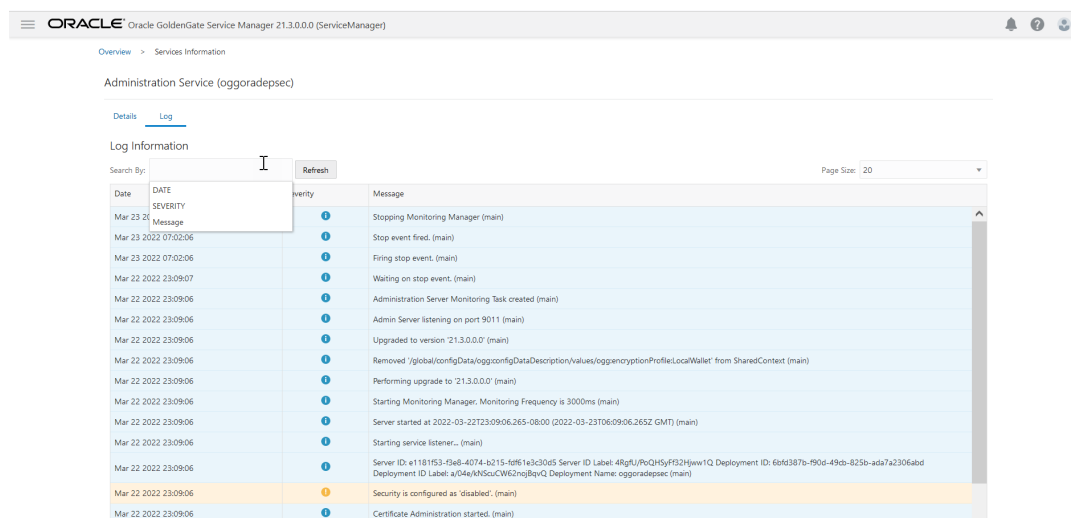
Search and Read Log Information for Microservices in a Deployment

You can view and search for log messages associated with one specific microservice in a deployment. This option narrows the log information to display the messages for the selected microservice in the deployment. To access the log information in this manner:

1. From the Services section of the Service Manager Overview page, click the **See Details** icon in the **Details** column.



2. Click the **Log** tab on the Service Information page. Log information specific to the microservice is displayed on this page.
3. Select the search criteria in the **Search** box to view specific log messages.



4. Enter values for the search criteria as discussed in [Search for Log Messages](#).

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.

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</code> with <code>"*"</code> removed
<code>INCOMING_REPLICAT</code>	=	<code>@GETENV ("GGENVIRONMENT", "GROUPNAME")</code>
<code>INCOMING_HEARTBEAT_TS</code>	=	<code>HEARTBEAT_TIMESTAMP</code>

```

INCOMING_EXTRACT_TS      = OUTGOING_EXTRACT_TS
INCOMING_ROUTING_TS      = OUTGOING_ROUTING_TS
INCOMING_REPLICAT_TS     = @DATE ('UYYYY-MM-DD
HH:MI:SS.FFFFFFF', 'JTSLECT', @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
INCOMING_EXTRACT	VARCHAR2	Name of the primary Extract (capture) at the remote database

Column	Data Type	Description
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	-
INCOMING_EXTRACT_RESTART_CSN	VARCHAR2	-

Column	Data Type	Description
INCOMING_EXTRACT_RESTART_TS	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 GG_HEARTBEAT_HISTORY 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 DB_NAME 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 SYSTIMESTAMP - REMOTE_EXTRACT_RESTART_TIME.
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 DB_NAME and DB_UNIQUE_NAME will be same. In a switchover to standby scenario, the db_unique_name will change but the db_name 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_TIME	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: SYSTIMESTAMP - REMOTE_EXTRACT_RESTART_TIME
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_REPICAT_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 PostgreSQL, you can set up an automated task to update the heartbeat table. See [Configure Automation Task to Update Heartbeat Table for PostgreSQL](#).

For all other supported databases, you must create background jobs to update the heartbeat timestamp using the database specific scheduler functionality.

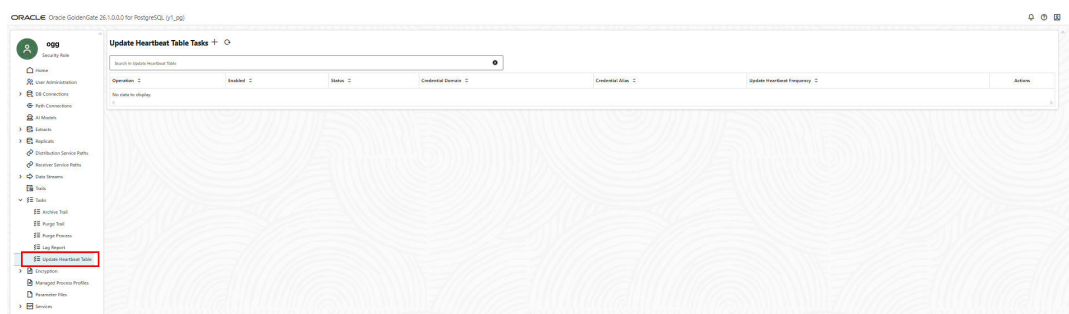
See ADD HEARTBEATTABLE, ALTER HEARTBEATTABLE for details on updating the heartbeat table.

Configure Automation Task to Update Heartbeat Table for PostgreSQL

The Oracle GoldenGate for PostgreSQL web interface provides the functionality to create a cron job for periodically updating the heartbeat table.

Following are the steps to create this task from the web interface:

1. From the left-navigation pane, expand **Tasks** and select the **Update Heartbeat Tasks** option.



2. Click the plus sign on the Update Heartbeat Tasks page. The **Create a New Heartbeat Task** dialog box is displayed.

Create a new Update Heartbeat Table task

Create a task to periodically update the heartbeat table by specifying the database credentials and the update frequency.

Name Enabled ☒

Domain Alias

Update Heartbeat Frequency Amount

Submit

3. Specify the following values in the dialog box and click **Submit**:
 - Name: Specify a name for the task.

- **Credential Domain:** Select the domain name from the drop down list.
- **Credential Alias:** Select the credential alias for the PostgreSQL database instance for which you are creating the update heartbeat task.
- **Frequency Interval:** Specify the frequency interval for running the update task. The minimum value is 60 (default) and the maximum value can be set to 7999.

To modify or delete an Update Heartbeat task, you can use the following steps:

1. From the left-navigation pane of the Administration Service interface, select **Tasks**.
2. Click **Update Heartbeat Tasks**. A list of existing Update Heartbeat Tasks is displayed.
3. From the **Actions** column, click the **pencil** icon. The **Edit Update Heartbeat Task** dialog box is displayed.
4. You can change or modify, the credential domain, alias, or the frequency interval and click **Submit**. The modified Update Heartbeat Task is listed on the Update Heartbeat Tasks page.
5. To delete a task, click the **Delete** icon from the Actions column for the task. Click OK to confirm deletion.

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 make 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.

Command	Description
DELETE HEARTBEATENTRY	Deletes entries in the heartbeat table.
INFO HEARTBEATTABLE	Displays heartbeat table information.

Reporting Lag

Lag reports can be accessed from the Administration Service. You can set up a Lag Report task to automatically generate lag reports for a certain tenure and issue warnings if the lag value crosses the specified threshold. To set up an automated task for generating lag reports:

1. From the Administration Service left navigation pane, click **Tasks** and select **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 generate a lag report automatically if the specified threshold is exceeded. You can specify the threshold value in the If Exceeds box.
 - If Exceeds: To specify a threshold after which a warning would be initiated. The value can be seconds, minutes, or hours.
 - Warning: To allow a warning to be generated in case the lag threshold exceeds the specified threshold.
 - When Exceeds: The lag threshold after which the warning is triggered.
5. Click **Submit**.

Identifying Lag in Replicat

To identify lag in Replicat process and modify columns of a table:

1. Suspend activity on the source database for all tables that are linked to Oracle GoldenGate.
2. Start Admin Client.
3. Run this command for the Replicat group:
`INFO REPLICAT group`
4. On the Checkpoint Lag line, verify whether there is any Replicat lag. If needed, continue to issue `INFO REPLICAT` until lag is zero, which indicates that all of the data in the trail has been processed.
5. Stop the Replicat group.
`STOP REPLICAT group`
6. Perform the table modifications on the target databases.
7. Start the Replicat process.
`START REPLICAT group`

8. Allow user activity to resume on all of the source tables that are linked to Oracle GoldenGate.

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 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

Table 10-1 (Cont.) Current Extract and Aged Reports

Permissions	X	Date	Report
-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 `TCPSSOURCETIMER` | `NOTCPSSOURCETIMER` 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 more about recommended procedures for improving the performance and error handling capabilities of Oracle GoldenGate components and processes.

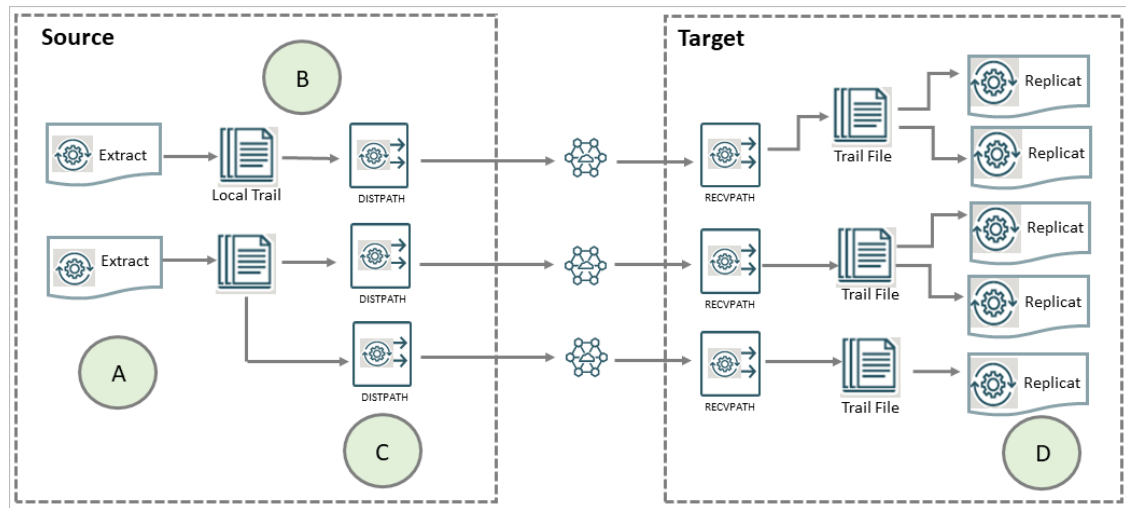
Using Multiple Process Groups

Typically, only one Extract group is required to efficiently capture from a database. However, depending on the redo (transactional) values, or the data and operation types, you may find that you are required to add one or more Extract group to the configuration.

Similarly, only one Replicat group is typically needed to apply data to a target database if using Replicat in coordinated mode. However, even in some cases when using Replicat in coordinated mode, you may be required to use multiple Replicat groups. If you are using Replicat in classic mode and your applications generate a high transaction volume, you probably will need to use parallel Replicat groups.

Each Oracle GoldenGate component, Extract, trail, `DISTPATH`, and Replicat, is an independent module, so you can combine them in ways that suit your needs. You can use multiple trails and parallel Extract and Replicat processes to handle large transaction volume, improve performance, eliminate bottlenecks, reduce latency, or isolate the processing of specific data.

The diagram below shows some of the ways that you can configure Oracle GoldenGate to improve throughput speed and overcome network bandwidth issues.



The image labels imply the following:

- **A:** Parallel Extracts divide the load. For example, by schema or to isolate tables that generate fetches.
- **B:** An Extract with local trail can be used for filtering, conversion, and network false tolerance.
- **C:** Multiple DISTPATHS work around network per-process bandwidth limitations to enable TCP/IP throughput. Divide the TABLE parameter statements among them.
- **D:** Parallel Replicats increase throughput to the database. Any trail can be read by one or more Replicats. Divide MAP statements among them.

Considerations for Using Multiple Process Groups

Before configuring multiple processing groups, review the following considerations to ensure that your configuration produces the desired results and maintains data integrity.

Maintaining Data Integrity

Not all workloads can be partitioned across multiple groups and still preserve the original transaction atomicity. You must determine whether the objects in one group will ever have dependencies on objects in any other group, transactional or otherwise. 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, you may be able to use multiple processes. Keep related DML together in the same process stream to ensure data integrity.

Number of Groups

The number of concurrent Extract and Replicat process groups that can run on a system depends on how much system memory is available. Each Extract and classic Replicat process needs approximately 25-55 MB of memory or more, depending on the size of the transactions and the number of concurrent transactions. The Oracle GoldenGate command interface fully supports up to 5,000 concurrent Extract and Replicat groups (combined) per instance of Oracle

GoldenGate. At the supported level, all groups can be controlled and viewed in full with commands such as the INFO and STATUS commands.

Beyond the supported level, group information is not displayed and errors may occur. Oracle GoldenGate recommends keeping the number of Extract and Replicat groups (combined) at a more manageable level, such as 100 or below, in order to manage the environment effectively. The maximum number of groups is controlled by the MAXGROUPS parameter, which has a default value of 1000.

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, but as you approach larger amounts of process groups, more than 60 or so, you will either need to adjust the 'non-interactive' value of the SharedSection field in the registry, based on this information from Microsoft (Windows desktop heap memory), or increase the number of Oracle GoldenGate homes and spread the total number of desired process groups across these homes.

Note

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

Memory

The system must have sufficient swap space for each Oracle GoldenGate Extract and Replicat process that will be running. To determine the required swap space:

1. Start up one Extract or Replicat.
2. Run Admin Client.
3. View the report file and find the line `PROCESS VM AVAIL FROM OS (min)`.
4. Round up the value to the next full gigabyte if needed. For example, round up 1.76GB to 2 GB.
5. Multiply that value by the number of Extract and Replicat processes that will be running. The result is the maximum amount of swap space that could be required

See the CACHEMGR parameter in *Parameters and Functions Reference for Oracle GoldenGate* for more information about how memory is managed.

Isolating Processing-Intensive Tables

You can use multiple process groups to support certain kinds of tables that tend to interfere with normal processing and cause latency to build on the target. For example:

- Extract may need to perform a fetch from the database because of the data type of the column, because of parameter specifications, or to perform SQL procedures. When data must be fetched from the database, it affects the performance of Extract. You can get fetch statistics from the STATS EXTRACT command if you include the STATOPTIONS REPORTFETCH parameter in the Extract parameter file. You can then isolate those tables into their own Extract groups, assuming that transactional integrity can be maintained.
- In its classic mode, Replicat process can be a source of performance bottlenecks because it is a single-threaded process that applies operations one at a time by using regular SQL. Even with BATCHSQL enabled (see *Parameters and Functions Reference for Oracle*

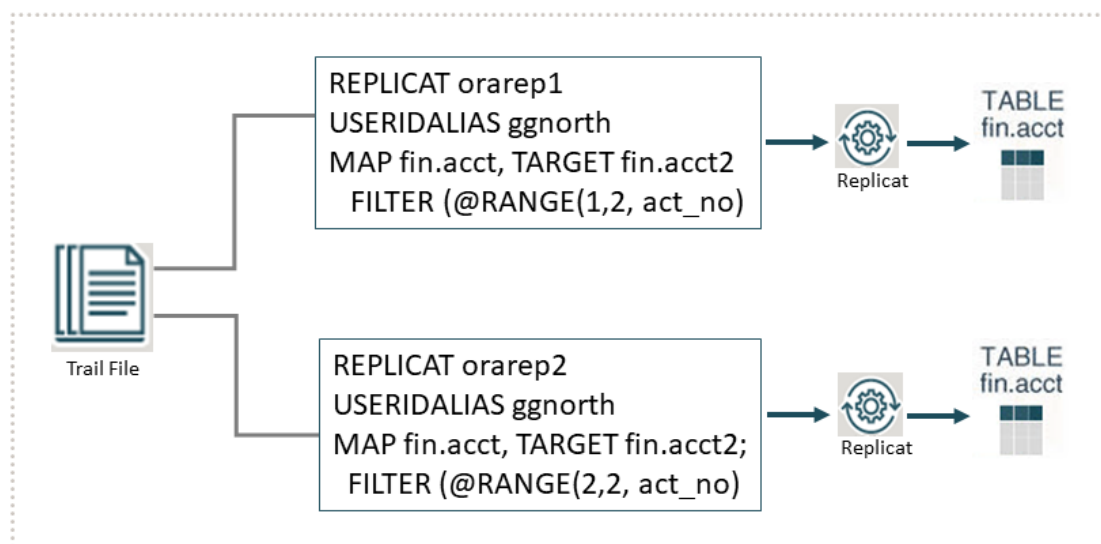
GoldenGate) Replicat may take longer to process tables that have large or long-running transactions, heavy volume, a very large number of columns that change, and LOB data. You can then isolate those tables into their own Replicat groups, assuming that transactional integrity can be maintained.

Splitting Large Tables Into Row Ranges Across Process Groups

You can use the @RANGE function to divide the rows of any table across two or more Oracle GoldenGate processes. It can be used to increase the throughput of large and heavily accessed tables and also can be used to divide data into sets for distribution to different destinations. Specify each range in a FILTER clause in a TABLE or MAP statement.

@RANGE is safe and scalable. It preserves data integrity by guaranteeing that the same row will always be processed by the same process group.

To calculate ranges, Replicat must filter through the entire trail to find data that meets the range specification. However, your business case should determine where this filtering is performed.



Eliminating Disk I/O Bottlenecks

I/O activity can cause bottlenecks for both Extract and Replicat.

- A regular Extract generates disk writes to a trail and disk reads from a data source.
- Each process writes a recovery checkpoint to its checkpoint file on a regular schedule.

Improving I/O performance Within the System Configuration

If there are I/O waits on the disk subsystems that contain the trail files, put the trails on the fastest disk controller possible.

Check the RAID configuration. Because Oracle GoldenGate writes data sequentially, RAID 0+1 (striping and mirroring) is a better choice than RAID 5, which uses checksums that slow down I/O and are not necessary for these types of files.

Improving I/O Performance Within the Oracle GoldenGate Configuration

You can improve I/O performance by making configurations changes within Oracle GoldenGate. Try increasing the values of the following parameters.

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

Note

CHECKPOINTSECS is not valid for an integrated Replicat on an Oracle database system.

- Use the GROUPTRANSOPS parameter to control the number of SQL operations that are contained in a Replicat transaction when operating in its normal mode. Increasing the number of operations in a Replicat transaction improves the performance of Oracle GoldenGate by reducing the number of transactions executed by Replicat, and by reducing I/O activity to the checkpoint file and the checkpoint table, if used. Replicat issues a checkpoint whenever it applies a transaction to the target, in addition to its scheduled checkpoints.

Note

GROUPTRANSOPS is not valid for an integrated Replicat on an Oracle database system, unless the inbound server parameter `parallelism` is set to 1.

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

Note

Increasing the values of these parameters improves performance, but it also increases the amount of data that must be reprocessed if the process fails. This has an effect on overall latency between source and target. Some testing will help you determine the optimal balance between recovery and performance.

Managing Virtual Memory and Paging

Because Oracle GoldenGate replicates only committed transactions, it stores the operations of each transaction in a managed virtual-memory pool known as a *cache* until it receives either a commit or a rollback for that transaction. One global cache operates as a shared resource of an Extract or Replicat process. The Oracle GoldenGate cache manager takes advantage of the memory management functions of the operating system to ensure that Oracle GoldenGate processes work in a sustained and efficient manner. The CACHEMGR parameter controls the amount of virtual memory and temporary disk space that is available for caching uncommitted transaction data that is being processed by Oracle GoldenGate.

When a process starts, the cache manager checks the availability of resources for virtual memory, as shown in the following example:

```
CACHEMGR virtual memory values (may have been adjusted)CACHESIZE: 32GCACHEPAGEOUTSIZE  
(normal): 8M PROCESS VM AVAIL FROM OS (min): 63.97GCACHESIZEMAX (strict force to disk):  
48G
```

If the current resources are not sufficient, a message like the following may be returned:

```
2025-11-11 14:16:22 WARNING OGG-01842 CACHESIZE PER DYNAMIC DETERMINATION (32G) LESS  
THAN RECOMMENDED: 64G (64bit system)vm found: 63.97GCheck swap space. Recommended swap/  
extract: 128G (64bit system).
```

If the system exhibits excessive paging and the performance of critical processes is affected, you can reduce the CACHESIZE option of the CACHEMGR. parameter. You can also control the maximum amount of disk space that can be allocated to the swap directory with the CACHEDIRECTORY option. For more information about CACHEMGR in the *Parameters and Functions Reference for Oracle GoldenGate*.

Tuning Replicat Transactions

Replicat uses regular SQL, so its performance depends on the performance of the target database and the type of SQL that is being applied (inserts, versus updates or deletes). However, you can take certain steps to maximize Replicat efficiency.

Tuning Coordination Performance Against Barrier Transactions

In a coordinated Replicat configuration, barrier transactions such as updates to the primary key cause an increased number of commits to the database, and they interrupt the benefit of the GROUPTRANSOPS feature of Replicat. When there is a high number of barrier transactions in the overall workload of the coordinated Replicat, using a high number of threads can actually degrade Replicat performance.

To maintain high performance when large numbers of barrier transactions are expected, you can do the following:

- Reduce the number of active threads in the group. This reduces the overall number of commits that Replicat performs.
- Move the tables that account for the majority of the barrier transactions, and any tables with which they have dependencies, to a separate coordinated Replicat group that has a small number of threads. Keep the tables that have minimal barrier transactions in the original Replicat group with the higher number of threads, so that parallel performance is maintained without interruption by barrier transactions.
- (Oracle RAC) In a new Replicat configuration, you can increase the PCTFREE attribute of the Replicat checkpoint table. However, this must be done before Replicat is started for the first time. The recommended value of PCTFREE is 90.

Preventing Full Table Scans in the Absence of Keys

If a target table does not have a primary key, a unique key, or a unique index, Replicat uses all of the columns to build its WHERE clause. This is, essentially, a full table scan.

To make row selection more efficient, use a KEYCOLS clause in the TABLE and MAP statements to identify one or more columns as unique. Replicat will use the specified columns as a key. The following example shows a KEYCOLS clause in a TABLE statement:

```
TABLE hr.emp, KEYCOLS (FIRST_NAME, LAST_NAME, DOB, ID_NO);
```


For usage guidelines and syntax, see the TABLE and MAP parameters in *Parameters and Functions Reference for Oracle GoldenGate*.

Splitting Large Transactions

If the target database cannot handle large transactions from the source database, you can split them into a series of smaller ones by using the Replicat parameter MAXTRANSOPS. See *Parameters and Functions Reference for Oracle GoldenGate* for more information.

Note

MAXTRANSOPS is not valid for an integrated Replicat on an Oracle database system.

Adjusting Open Cursors

The Replicat process maintains cursors for cached SQL statements and for SQLEXEC operations. Without enough cursors, Replicat must age more statements. By default, Replicat maintains as many cursors as allowed by the MAXSQLSTATEMENTS parameter. You might find that the value of this parameter needs to be increased. If so, you might also need to adjust the maximum number of open cursors that are permitted by the database. See *Parameters and Functions Reference for Oracle GoldenGate* for more information.

Improving Update Speed

Excessive block fragmentation causes Replicat to apply SQL statements at a slower than normal speed. Reorganize heavily fragmented tables, and then stop and start Replicat to register the new object ID.

Set a Replicat Transaction Timeout

Use the TRANSACTIONTIMEOUT parameter to prevent an uncommitted Replicat target transaction from holding locks on the target database and consuming its resources unnecessarily. You can change the value of this parameter so that Replicat can work within existing application timeouts and other database requirements on the target.

TRANSACTIONTIMEOUT limits the amount of time that Replicat can hold a target transaction open if it has not received the end-of-transaction record for the last source transaction in that transaction. By default, Replicat groups multiple source transactions into one target transaction to improve performance, but it will not commit a partial source transaction and will wait indefinitely for that last record. The Replicat parameter GROUPTRANSOPS controls the minimum size of a grouped target transaction.

The following events could last long enough to trigger TRANSACTIONTIMEOUT:

- Network problems prevent trail data from being delivered to the target system.
- Running out of disk space on any system, preventing trail data from being written.
- Extract abends or is terminated in the middle of writing records for a transaction.
- There is a source system failure, such as a power outage or system crash.

See TRANSACTIONTIMEOUT in *Parameters and Functions Reference for Oracle GoldenGate* for more information.

Using Healthcheck Scripts to Monitor and Troubleshoot

Oracle GoldenGate Healthcheck script provides database site information for Oracle Databases to allow monitoring and troubleshooting.

The Healthcheck script gathers all replication related configuration and performance information from a database in one single run. Within the scripts, there are many queries regarding the database instance and the database specific information from Extract and Replicat. You can run the script periodically to obtain the latest database side performance information regarding replication.

The output is one of the key information that is needed for support for a qualitative analysis of the replication environment.

Installing, Running, and Uninstalling Healthcheck Scripts

The Healthcheck directory contains three files to install, run and deinstall the Healthcheck script. After the PL/SQL package is installed with `ogghc_install.sql`, you can frequently run the `ogghc_run.sql` script to generate an output file.

You can deinstall the Healthcheck script using the `ogghc_deinstall.sql` script.

The Healthcheck script is located in:

`$OGG_HOME/lib/sql/healthcheck`

To gather information, Oracle recommends you to install and run the Healthcheck as a SYS user. However, you can also install and run the script as an Oracle GoldenGate administrator user role. In this case, some system information is not available. The Healthcheck output displays the information that requires SYS privileges.

How to Deal with Healthcheck Information?

The output file of the Healthcheck script contains the instance name and a timestamp. By default, information about Integrated Extract and Replicat is gathered. However, you can retrieve information from the legacy Oracle GoldenGate schema or database profile. For this reason, you have to take out the argument of the EXCLUDETAG parameter.

Depending of the amount of information being queried, the run time of the script varies (in minutes).

You can eliminate a query that takes too long to process using the Healthcheck script and run another query in a parallel session to get the output.

Components of Healthcheck Information

The Healthcheck script generates an HTML file with JSON objects and HTML code. The output of the Healthcheck script contains the following sections:

- Overview
- Extract
- Replicat
- Table Statistics and Errors

- Tools
- Report Map (Legacy)

Each of these sections contain menus and sub-menus depending on the type of data available.

The following table describes the sections and the data available in those sections based on the query used to generate the Healthcheck output.

Menu	Description
Overview	The Overview section contains information about the following: • General Findings • Database, Extract and Replicat Summary • Capture Parameters • Apply Parameters
Database	Main Menu (The Main menu already contains the query information)
Database Objects	This sub-menu option displays information about the following: • Tables Not Supported By Oracle GoldenGate Integrated Capture • Instantiation SCNs for Apply Schema and Database (DDL)
Database Details	This sub-menu has a detailed database related various connections and services along with key Oracle GoldenGate parameters and the manual or modified database parameters. The second part shows the basic information about the components, software and patch level of the database. The information is distributed amongst the following sections. • Connection Information • Key init.ora parameters • Database dictionary and fixed table statistics • Software edition • Registry information including History • Replication bundled patch information • NLS Database parameters • Registered log files for capture • Current Database incarnation • Standby redo logs • GoldenGate Administrator User
Replication SQL Analysis	This section provides a complete log of Oracle GoldenGate related information from the session at run time of the script and active session history. This contains complete details about Waits, Events, IO, Contention and SQL.
Objects Instantiation	This sub-menu provides details about the schemas and table-level supplemental logging for Oracle GoldenGate: • Schemas prepared for capture • Table-level supplemental log groups enabled for capture
Extract	Main Menu
Extract Details	The Extract details covered in this sub-menu are: • Capture Runtime Information • Capture Transaction Processing • Capture Processing Information • Logminer Session Statistics • Capture Rules & Rulesets

Menu	Description
Extract Performance	This sub-menu displays information about the Extract performance depending on the type of Extract being used. It displays the progress of the Extract which includes the following details: • Capture Name - Name of the Extract • Client Name - Name of the client where the Extract is running • Client Status - Status of the client where the Extract is running • Processed Low SCN - Processed SCN value of the Extract • Oldest SCN - Oldest SCN value of the Extract
Extract Logminer	This sub-menu contains information mainly used for debugging issues.
Replicat	Main Menu
Replicat Performance	The Replicat performance provides the following information: • Apply Progress • GoldenGate Inbound Progress Table • Information about Apply Progress table • Apply Network Receiver (ANR) • Apply Reader • Apply Reader - Dequeue Information • Apply Coordinator • Apply Coordinator Watermarks • Open GoldenGate Apply Transactions • Open GoldenGate Apply Transactions -Details • Apply Server Transactions ordered by Server_id • Apply Server Statistics - Summary • Apply Server Statistics - Details • Apply Server Statistics - Auto Tuning • Apply Server Wait Events • Apply Server Session Events • Apply Reader Processes • Apply Coordinator Processes • Apply Server Processes
CDR	This sub-menu provides a detailed log of the Replicat error handlers.
Apply Handler	This sub-menu contains information about the Replicat name, DDL handler, and precommit handler.
Error Management Details	This sub-menu has details about the Oracle GoldenGate table statistics sorted by table. It includes information such as server name, source table owner, source table name, destination table owner, destination table name, total operations, inserts, updates, deletes, insert/update/delete collisions, REPERERROR discards, REPERERROR ignores, WAIT dependencies, and CDR related updates.
Tools	Main Menu
History	This sub-menu is dependent on the type of query you have run and displays subscriber history, Extract, and Replicat history.

Menu	Description
Report Map	Main Menu (Legacy). It provides information on all the queries, which includes details such as: • Run time of the query • Number of Returned Rows • Information if the query has succeeded or failed • Information if this is internal information that is only visible if the script is run as SYS • Disabled Queries The hyperlinks directs you to the appropriate query.
Hints/Description	This sub-menu is a map of all the activities logged in the Healthcheck report.
Truncates	This sub-menu displays the Oracle GoldenGate related information from the V\$ACTIVE_SESSSION_HISTORY and V\$SQLAREA database views.
 Note You can view only partial results for V\$ACTIVE_SESSSION_HISTORY , as the original size of the query exceeds the maximum limit.	
Config	This sub-menu allows you to add rules to the following sections that are available on this page: • Column Rules: There can be any number of rules on a single table and they will be applied one after the other in the order they appear in the rules table. It can be an expression using the values of the columns in a row. • Menu Items Exclusions: This option is used to exclude any menu item from the Healthcheck output. Click Add to set the rule. • Group By: This section is used to group by keys and aggregates. You can go to any statistic and click G to add a group by sorting.
JS Errors	Main Menu. This page displays debugging information for this framework.
 Note It is visible only in case of errors.	

11

Mission Critical

Learn about Oracle GoldenGate features that provide mission critical support for varied environments.

Configuration Service

Starting with Oracle GoldenGate 23ai, a new Configuration Service is available for managing the configuration files of Oracle GoldenGate. The Configuration Service provides an alternate method for storing checkpoint and recovery information. This alternate method uses a database repository instead of the file system. A separate ConfigService program within the Service Manager deployment, manages this service.

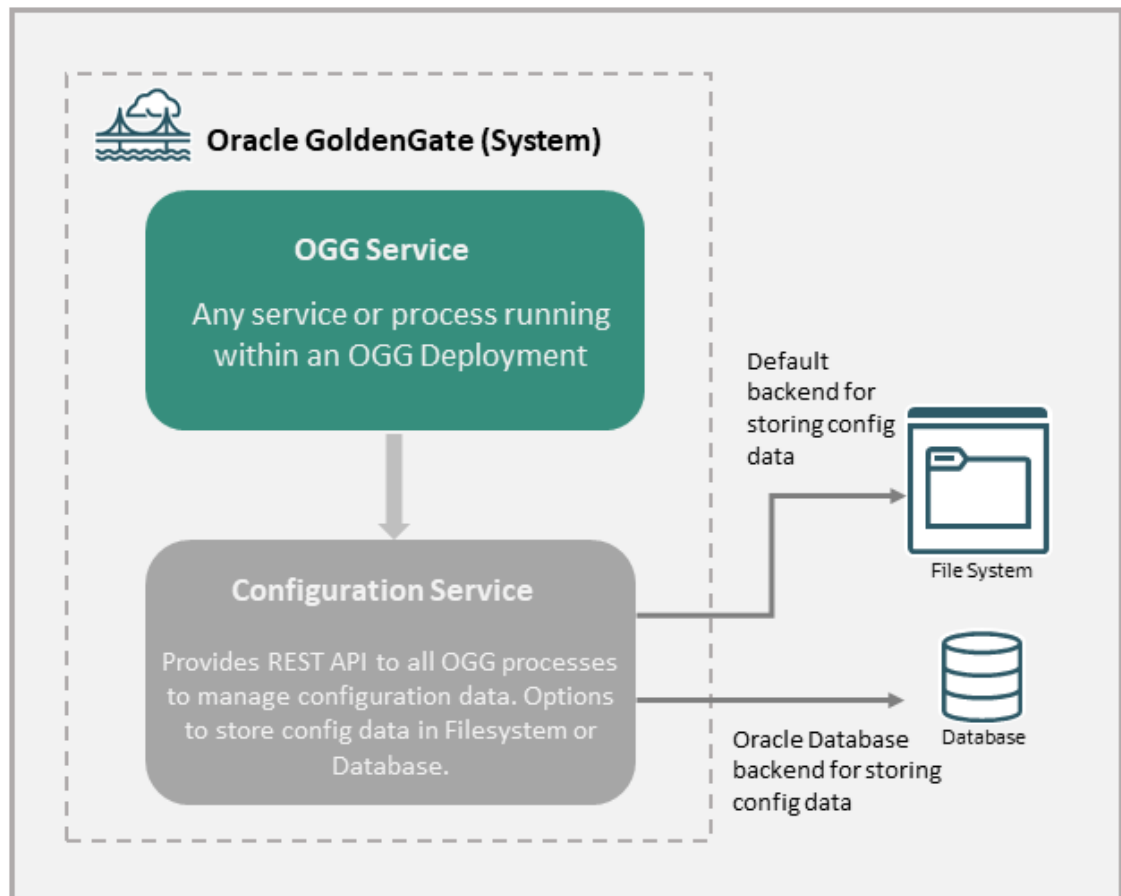
Note

The Configuration Service is available with Oracle database.

The ConfigService manages configuration files from all deployments that are managed by this Service Manager. Configuration data includes:

- Microservices configuration data
- Parameter files
- Checkpoint data
- Database and user credentials

When you enable the ConfigService, this configuration data be stored seperately on another file system or a database as a backend, as shown in the following diagram.



Here are the **features** of the Configuration Service:

- The Configuration Service provides a REST API to all other processes with an Oracle GoldenGate installation to manage critical configuration data.
- Multiple Oracle GoldenGate installations can use the same Data Store.
 - The Configuration Data Store is managed independently of Oracle GoldenGate.
 - External applications can query configuration data independently of Oracle GoldenGate.
- Transparent failover and switching to a different server is supported.

Consider a situation where two identical Oracle GoldenGate installations on different nodes use the Configuration Service to store configuration data externally in a database server for high availability. In case one of the installations experiences a failure, then using the failover mechanism, it automatically switches from one node to the other, and continues to use the Configuration Service to transmit and store configuration data to the database server.

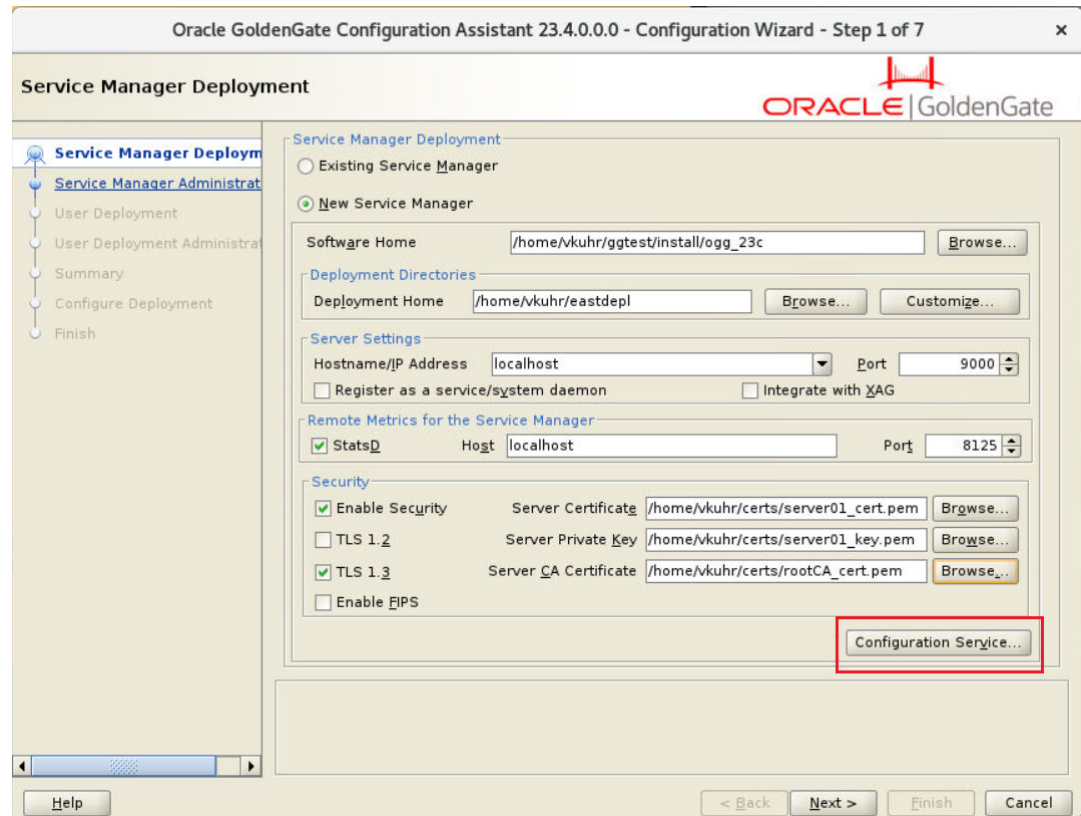
- When you choose to use the Configuration Service, the configuration files are not stored in \$OGG_ETC_HOME or \$OGG_VAR_HOME directories.

Enable the Configuration Service

Configuration Service can be enabled using the OGGCA utility.

To enable the Configuration Service using OGGCA, use the following steps:

1. Run the OGGCA utility and after setting the other options for the Service Manager, select the Configuration Service button, as shown in the following image:

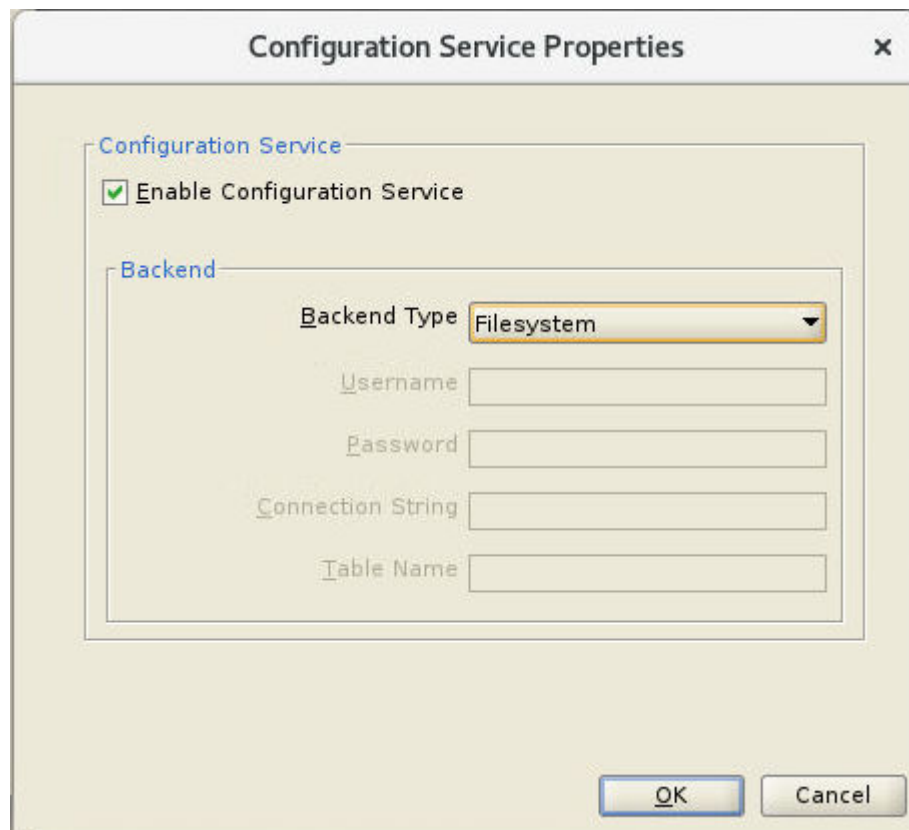


2. When you click the Configuration Service button, the Configuration Service dialog box appears.

Note

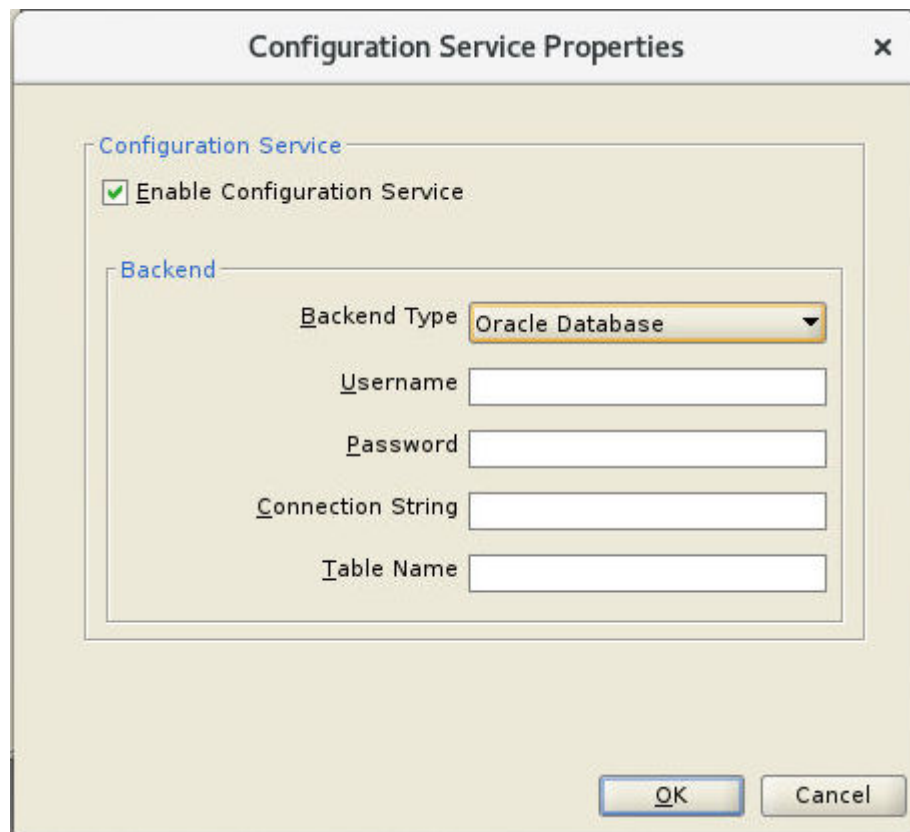
The options in the Configuration Service dialog box are disabled for non-Oracle databases.

3. Select the **Enable Configuration Service** check box and set up the back end options. The default option is **Filesystem**. You can also choose to use an Oracle Database.



The image shows a dialog box titled "Configuration Service Properties" with a close button (X) in the top right corner. The dialog is divided into two sections. The first section, "Configuration Service", contains a checked checkbox labeled "Enable Configuration Service". The second section, "Backend", contains a dropdown menu for "Backend Type" with "Filesystem" selected. Below the dropdown are four text input fields: "Username", "Password", "Connection String", and "Table Name". At the bottom right of the dialog are "OK" and "Cancel" buttons.

4. If you select the **Oracle Database** option, specify the login credentials and connection details for the database server, shown in the following image.

The image shows a 'Configuration Service Properties' dialog box. It has a title bar with the text 'Configuration Service Properties' and a close button (X). The main area is divided into two sections. The first section, 'Configuration Service', contains a checked checkbox labeled 'Enable Configuration Service'. The second section, 'Backend', contains a dropdown menu for 'Backend Type' with 'Oracle Database' selected. Below this are four text input fields: 'Username', 'Password', 'Connection String', and 'Table Name'. At the bottom right are 'OK' and 'Cancel' buttons.

If you choose the Oracle Database option, you can either use the source/target database being used for data replication or a different database outside the replication. Configure the following options for Oracle database:

- Username: The username of the database user.
- Password: Password for the database user login.
- Connection String: The URL used to connect to the database. This is the backend connection string. The connection string can use the data source in the format: host[:port]/service_name or a TNS_ALIAS.

For example, *localhost:1523/cdb1_pdb1.rdbms.oracle.com* is an example of **hostname**, **port number**, and **service name** format.

- Table Name: Database backend table name where the configuration files would be stored. The backend table might be in any schema. In this example, the Oracle GoldenGate Admin schema (ggadmin) is used.

Example:

ggadmin.ggs_backendtable.

For details about the OGGCA utility, see [Add a Deployment](#).

Configuration Service: Considerations

Consider the following guidelines before choosing to use the Configuration Service (ConfigService) in OGGCA:

- ConfigService is configured when adding deployments using OGGCA. See [Select Service Manager Options](#).

- The **Database** and **Fileset** options both provide high availability with Oracle GoldenGate.
- If you choose to use the ConfigService, then you cannot revert to using conventional File Management system.
- If you select the **Database** option, then use the following guidelines:
 - The database used by the ConfigService can be in a Real Application Cluster and/or a Data Guard environment.
 - The database backend only supports Oracle database.
 - You can query the backend table of the ConfigService, but you cannot modify its content.
 - The tablespace that contains the backend table must use a locally managed tablespace.
 - The backend table can be at a remote database.
 - As many Configuration Services from different deployments can use one and the same backend table, the configuration data can be centralized.

Backing up the Oracle GoldenGate Environment

After you start Oracle GoldenGate processing, an effective backup routine is critical to preserving the state of processing in the event of a failure. Unless the Oracle GoldenGate working files can be restored, the entire replication environment must be re-instantiated, complete with new initial loads.

As a best practice, include the entire Oracle GoldenGate home installation (\$OGG_HOME) in your backup routines. There are too many critical sub-directories, as well as files and programs at the root of the directory, to keep track of separately. In any event, the most critical files are those that consume the vast majority of backup space, and therefore it makes sense just to back up the entire installation directory for fast, simple recovery. Apart from Oracle GoldenGate home installation directory, you must back up the \$OGG_ETC_HOME directory, which contains the configuration settings for the Oracle GoldenGate installation.

12

Quickstarts

This section lists the Oracle GoldenGate quickstarts.

Switching from Nonintegrated Replicat to Parallel Nonintegrated Replicat

The process for switching to *parallel integrated* or *parallel nonintegrated Replicat* is the same for all Replicat modes. This topic describes the process to *switch from nonintegrated Replicat to parallel nonintegrated Replicat*.

Before Starting the Switching Process

1. Create a parallel nonintegrated Replicat process, **repe** that reads from the existing trail file:

```
ADD REPLICAT repe, PARALLEL, EXTTRAIL ea, checkpointtable  
ggadmin.ggs_checkpoint1
```

In this command, **repe** is the name of the Replicat. **ea** is the trail name. The trail name supplied while creating this Replicat is the same as the other Replicat in nonintegrated mode.

Note

If the checkpoint table is configured in GLOBALS, then there is no need to include the checkpoint table option with this command. If not, then use this option to provide the checkpoint table name.

2. Do **not** start the parallel nonintegrated Replicat (repe).
3. Stop the current nonintegrated Replicat, **repea**.

```
STOP REPLICAT repea
```

4. On the target side, access the Replicat report file (.rpt) to know the values of the following components:
 - Last applied CSN by the current nonintegrated Replicat process.
 - Trail sequence and RBA of the existing Replicat process.

To access the details of the Replicat, run the command:

```
INFO REPLICAT repea DETAIL
```

The output for this command shows an output similar to the following:

```

Replicat   REPEA      Last Started 2022-06-16 04:21   Status STOPPED
Description      eastt
Checkpoint Lag   00:00:00 (updated 01:59:59 ago)
Log Read Checkpoint File east/ea0000000009
                2022-06-14 04:38:34.084220   RBA 9382
Settings Profile Default
Encryption Profile LocalWallet

```

Current Log BSN value: (no data)

Last Committed Transaction CSN value: 50698907

Extract Source	Begin	End
east/ea0000000009	2022-06-16 04:21	2022-06-14
04:38		
east/ea0000000000	* Initialized *	2022-06-16
04:21		
east/ea0000000009	2022-06-14 04:38	2022-06-14
04:38		
east/ea0000000009	2022-06-14 04:38	2022-06-14
04:38		
east/ea0000000009	2022-06-16 03:55	2022-06-14
04:38		
east/ea0000000000	* Initialized *	2022-06-16
03:55		
east/ea0000000000	* Initialized *	First
Record		
east/ea0000000000	* Initialized *	First
Record		
east/ea0000000000	* Initialized *	First
Record		
east/ea0000000000	* Initialized *	First
Record		

Current directory /scratch/preeshuk/ggtest/install_ogg21.3_210725/bin

```

Report file      /scratch/oggoradep/var/lib/report/REPEA.rpt
Parameter file   /scratch/oggoradep/etc/conf/ogg/REPEA.prm
Checkpoint file   /scratch/oggoradep/var/lib/checkpt/REPEA.cpr
Checkpoint table  DBEAST.GGADMIN.GGS_CHECKPOINT
Process file      /scratch/oggoradep/var/run/REPEA.pcr
Error log        /scratch/oggoradep/var/log/ggserr.log

```

Start the Switching Process

To start using the nonintegrated parallel Replicat, you need to alter it to port the content from the other Replicat. Use the following steps to perform this task:

1. Run the ALTER REPLICAT command as follows:

```
ALTER REPLICAT replicat_name, EXTSEQNO extseqno, EXTRBA extrba
```

For example, for the Replicat repe, here's the command:

```
ALTER REPLICAT repe, EXTSEQNO 9, EXTRBAm 9382
```

2. Start the newly created parallel nonintegrated Replicat process using the following command:

```
START REPLICAT repe AFTERCSN csn_value
```

For example:

```
START REPLICAT repe AFTERCSN 50698907
```

This starts the Replicat at the specified CSN value in the trail file.

Connecting Two Deployments Using External RootCA Certificate

There are multiple approaches which you can implement for applying certificates when working across different source and target deployments.

This quickstart demonstrates how to set up and apply certificates when using **external RootCA** certificate.

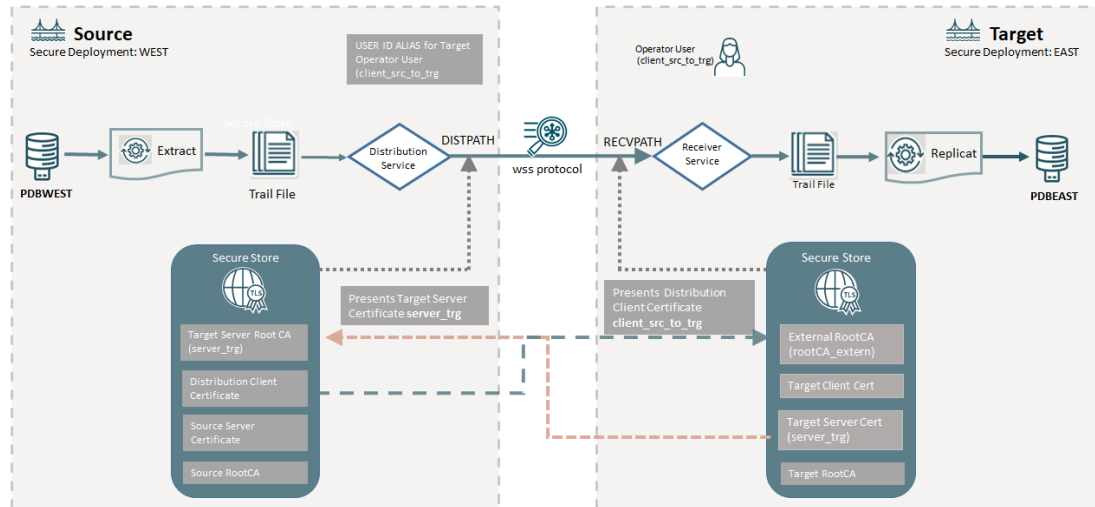
Environment

Each deployment uses its own set of Root, Server, and Client certificates generated for that system. These server and client certificates are imported at the time of configuring deployment with the OGGCA utility. As this quickstart assumes to use a secure deployment, the server certificates and the corresponding root certificates are already installed at the time of deployment. In this quickstart, you will learn how an independent (external) Client Certificate is added to the source deployment for authenticating the Distribution Path (using the wss protocol) on the target deployment.

- **Source:** *west.oracle.com*
- **Target:** *east.oracle.com*

The target server presents a Server Certificate to the source deployment. The pre-installed CA Certificate at the source verifies the identity of the target Server Certificate. Similarly, the source distribution client presents a Client Certificate to the target deployment and the pre-

installed CA Certificate on the target site verifies the identity of the distribution client.



The process includes the following steps:

1. Create an additional external distribution path client (**dist_client**) certificate signed by an external Certificate Authority (rootCA_extern) for the Distribution Path using the secure Web-Socket protocol (wss).
2. On the source deployment, apply the target server certificate (created for the initial deployment) as a root CA certificate. This allows the source deployment to validate the authenticity of the target system.
3. Integrate the external **dist_client** to the system:
 - a. In the source deployment, apply the external **dist_client** certificate.
 - b. In the target deployment, apply the external root CA certificate (rootCA_extern) from the external dist_client certificate.

Now, the target deployment can validate the authenticity of the external dist_client certificate.

4. In the target deployment, create an Oracle GoldenGate user certified by the **dist_client** certificate with the Operator role. This user automatically gets the name in form of a Common Name (CN).
5. In the source deployment, create the distribution path using the wss protocol with the **Certificate** target authentication method. This certificate matches the Oracle GoldenGate CN user at the target deployment.

Create the Distribution Client and External RootCA Certificates

This part of the quickstart is a prerequisite to the section [Connect the Two Deployments Using the Distribution Path](#).

Create the Distribution Path Client Certificate

The distribution path client certificate is an additional certificate required when using external trusted root CA certificate for authenticating a connection between the Distribution Path on the source deployment (west) with the external trusted root CA certificate (rootCA_extern) on the target deployment (east). So, creating the distribution path client (dist_client) certificate is a prerequisite. The other certificates for secure source and target deployments must be up and running already.

Follow the given steps, to create configuration file and certificates for the source and target deployment.

1. Create the rootCA_extrn certificate:

- a. Use a configuration file similar to the following for rootCA_extrn:

```
[ req ]
default_bits = 4096
default_md = sha512
prompt = no
encrypt_key = no
distinguished_name = req_distinguished_name
req_extensions = v3_req
x509_extensions = v3_ca
x509_extensions = usr_cert
[ req_distinguished_name ]
commonName = "rootCA_extrn"
[ v3_req ]
basicConstraints=CA:TRUE
[ v3_ca ]
basicConstraints=CA:TRUE
[ usr_cert ]
basicConstraints=CA:TRUE
[ my_extensions ]
EOF
```

- b. Use the following command to create the rootCA_extrn certificate:

```
# rootCA certificate
openssl req -x509 -newkey rsa:4096 -nodes \
            -keyout rootCA_extrn.key \
            -out rootCA_extrn.cert -days 73000 \
            -config rootCA_extrn.cfg
```

2. Create an external Distribution Path Client (dist_client) certificate. Create a **client_west_to_east.cfg** configuration file similar to the following:

- a. Create a **client_west_to_east.cfg** configuration file similar to the following:

```
[ req ]
default_bits = 4096
default_md = sha512
prompt = no
encrypt_key = no
distinguished_name = req_distinguished_name
[ req_distinguished_name ]
commonName = "client-west-to-east"
[ my_extensions ]
EOF
```

- b. Create the **client_west_to_east** distribution client certificate.

```
openssl req -new -newkey rsa:2048 -nodes \
            -keyout client_west_to_east.key \
```



```
-out client_west_to_east.csr \
-config client_west_to_east.cfg
```

This certificate is verified by the rootCA_extern certificate when the source distribution client (dist_client) connects to the target deployment (east).

```
openssl x509 -req -days 73000 \
-in client_west_to_east.csr \
-CA rootCA_extern.cert \
-CAkey rootCA_extern.key \
-CAcreateserial \
-out client_west_to_east.cert
```

Apply Certificates to Source and Target Deployments

Oracle GoldenGate provides two ways for applying certificates for deployments:

- Applying Certificates When Creating a Secure Deployment Using OGGCA: Use this option if you have existing wallets and certificates on source and target deployments.
- Applying Certificates Using the Service Manager Certificate Management page: Use this option when there are existing deployments, where certificates need to be applied on the source and target deployments.

In this quickstart, the Certificate Management method is used to apply certificates while setting up a secure deployment.

1. Log in to the Service Manager and navigate to the Certificate Management page.
2. At the source, add the target Server CA certificate (server_east).

The Distribution service on the source system must trust the target server certificate, which is authorized by the server_east target server CA certificate. This is added to the source secure store.

CA Certificates +						
Shared						
Status	Name	Issued To	Issued By	Validity Start	Validity Expiration	Action
✓	server.default-RootCA_west-RootCA_west	CN = RootCA_west	CN = RootCA_west	2021-07-02T16:04:08Z	2022-07-02T16:04:08Z	  
✓	server_east	CN = jflz	CN = RootCA_east	2021-07-02T16:37:37Z	2022-07-02T16:37:37Z	  

See [Manage Certificates for Deployments](#) to know the steps for accessing the Certificate Management page where you can add CA, Server, and Client certificates.

3. At the source, add a **Distribution Path Client Certificate**. The Distribution Path Client certificate is created in addition to the initial setup and is used connecting the Distribution Path to the target. This client certificate is signed by another trusted Root CA certificate (**rootCA_extern**), which is added to the target deployment. Both certificates are independent from the certificates from the initial deployment of the Oracle GoldenGate source and target instances.

Client Certificates +						
Status	Name	Issued To	Issued By	Validity Start	Validity Expiration	Action
✓	client_west_to_east	CN = client_west_to_east	CN = rootCA_extern	2021-07-02T17:27:03Z	2085-04-07T10:58:47Z	  
✓	default	CN = client_west	CN = RootCA_west	2021-07-02T16:04:22Z	2022-07-02T16:04:22Z	  

4. At the target, add the trusted Root certificate **rootCA_extern** for the client certificate **client_west_to_east**, which was added to the source deployment. The **rootCA_extern** certificate is added as the **CA Certificate** for **client_west_to_east** certificate. The Receiver Service on the target system must trust either the client certificate or the issuer of the client certificate. This needs to be added to the target secure store.

CA Certificates +

Shared

Status	Name	Issued To	Issued By	Validity Start	Validity Expiration	Action
✓	rootCA_extern	CN = rootCA_extern	CN = rootCA_extern	2021-07-02T17:27:02Z	2085-04-07T10:58:46Z	  
✓	server.default-RootCA_west-RootCA_west	CN = RootCA_TRG	CN = RootCA_east	2021-07-02T16:37:30Z	2022-07-02T16:37:30Z	  

5. Add the target certificate, **server_east** to the target deployment. This certificate is presented to the source deployment to make sure that the connected deployment is the correct target deployment. On the source side, the **server_east** certificate is verified by the **server_east** trusted CA certificate.

After the certificates are added on the source and target deployment, you can configure the distribution path on the source deployment to connect to the target.

Connect the Two Deployments Using the Distribution Path

After applying certificates provided in [Create the Distribution Client and External RootCA Certificates](#), perform the following steps to connect source and target deployments through the distribution path.

Set Up the Distribution Path between Source (west) and Target (east)

1. From the target deployment web interface, add a user with the Operator role, for connecting to the target deployment using the distribution path.
 - a. From the Administration Service, select User Administration.
 - b. Click the plus sign next to Users.
 - c. Specify the details for target deployment Operator user.

Notice that the username is set as the **CN** value mentioned in the distribution path client certificate.

Note

In this quickstart, the **Certificate** type of user authentication is used. You can also use the **Password** type for authentication.

- d. Upload the client certificate click **Submit**. The user is created.

Create new User



Create a new GoldenGate user identified by either a password or a client certificate. Also assign the appropriate security role to the new user

Authenticated By
Certificate

Role
Operator

Info

Certificate:

Upload

```
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBCwggSjAgEAA
oIBAQDjlGE4WHeTApeE
dXnZXInAhs/s9BRVimrdRJAzOHEm+7DjNjx5Bpn1nR6mZu
FotFeiYDeVW1fM4g4
```

CA Certificates:

Enter Upload Clear

Submit

2. Now, you can create and start the distribution path at the source system using the client certificate that was created for routing data from source to target. The following list describes the configuration of the Distribution Path with the Target Authentication Method set as **Certificate** and the Certificate value shows **client_west_to_east**.

The following values must be entered in the Add Distribution Path dialog box:

- 1: Target Authentication Method is selected as Certificate
- 2: Target protocol is selected as **wss**
- 3: Target name is the same as the CN name specified in the **client_west_to_east.cfg** file
- 4: Certificate name is the name of the Distribution Path client certificate, **client_west_to_east**.

You can also see [Add a Distribution Path](#) to know more about other options for configuring a distribution path.

13

Appendix

Learn about additional details required for supporting Oracle GoldenGate on different databases.

Sample Commands to Configure GoldenGate Data Streams for JSON Relational Duality Views

This section provides a sample workflow using cURL commands, to configure GoldenGate Data Streams for JSON Relational Duality Views.

```
#---
-----
-----
# --
# -- Create USERIDALIAS to connection from GoldenGate to the Databases
# --
#
-----
-----

curl -s -k -X POST https://north:9001/services/v2/credentials/
OracleGoldenGate/ggnorth \
-H "Content-Type: application/
json" \
-H "Accept: application/
json" \
-H 'Authorization: Basic
*****' \
-d '{"userid":"ggadmin@dbnorth","password":"****"}' | jq '.messages'

#
-----
-----
# --
# -- Add Schematranda/Trandata
# -- Note that the JSON DV and JCT are explicitly called for trandata
# --
#
-----
-----

curl -s -k -X POST https://north:9001/services/v2/connections/
OracleGoldenGate.ggnorth/trandata/schema \
-H "Content-Type: application/
json" \
```

```

-H "Accept: application/
json" \
-H 'Authorization: Basic
*****' \
-d '{"operation":"add","schemaName":"hr"}' | jq '.messages'

curl -s -k -X POST https://north:9001/services/v2/connections/
OracleGoldenGate.ggnorth/trandata/table \
-H "Content-Type: application/
json" \
-H "Accept: application/
json" \
-H 'Authorization: Basic
*****' \
-d '{"operation":"add", "tableName":"hr.students_dv"}' | jq '.response'

curl -s -k -X POST https://north:9001/services/v2/connections/
OracleGoldenGate.ggnorth/trandata/table \
-H "Content-Type: application/
json" \
-H "Accept: application/
json" \
-H 'Authorization: Basic
*****' \
-d '{"operation":"add", "tableName":"hr.demo_jct"}' | jq '.response'

#
-----
# --
# -- Add Extracts on source database GGNORTH
# -- Note that the wildcard within the TABLE parameter captures
# --
#
-----

curl -s -k -X POST https://north:9001/services/v2/extracts/
EXTN \
-H 'Content-Type: application/
json' \
-H 'Authorization: Basic
*****' \
-d '{"description":"Extract - Region North"
  , "config":["EXTRACT EXTN"
    , "USERIDALIAS ggnorth"
    , "EXTTRAIL north/ea"
    , "DDL INCLUDE MAPPED"
    , "DDLOPTIONS REPORT"
    , "REPORTCOUNT EVERY 10 MINUTES, RATE"
    , "WARNLONGTRANS 15MINUTES, CHECKINTERVAL 5MINUTES"
    , "TABLE hr.*;"
  ]
  , "source": "tranlogs"
  , "credentials":{"alias":"ggnorth"}
  , "registration": {"optimized": false}

```

```

        , "begin": "now"
        , "targets": [{"name": "ea", "path": "north/"}]
        , "status": "running"
    }' | jq '.messages'

#
-----
-----
# --
# -- Add User for DataStream
# --
#
-----
-----

curl -s -k -X POST https://north:9001/services/v2/authorizations/Operator/
dsadmin \
-H 'Content-Type: application/
json' \
-H "Accept: application/
json" \
-H 'Authorization: Basic
*****' \
-d '{"credential": "****", "info": "Dedicated DataStream user"}' | jq
'.response'

#
-----
-----
# --
# -- Add DataStream
# --
#
-----
-----

curl -s -k -X POST https://north:9002/services/v2/stream/
DS01 \
-H 'Content-Type: application/
json' \
-H "Accept: application/
json" \
-H 'Authorization: Basic
*****' \
-d '{"source": {"trail": "ea"
                , "path": "north"
                }
     , "cloudEventsFormat": false
     , "encoding": "json"
     , "bufferSize": 1048576
     , "qualityOfService": "exactlyOnce"
     , "description": "Data Stream Test #01"
    }' | jq '.response'

#

```

OGGCA Sample Template Response File for Silent Deployment

This section provides the sample template response file for starting a silent deployment using the OGGCA utility.

OGGCA Sample Template Response File for Silent Deployment

You can use the following sample template response file for starting a silent deployment using the OGGCA utility.

```
#####  
##  
##  
##  
## Oracle GoldenGate deployment configuration options and details  
##  
##  
##  
#####  
##  
  
#####  
##  
##  
##  
## Instructions to fill out this response file  
##  
## -----  
##  
## Fill out section A, B, and C for general deployment information  
##  
## Additionally:  
##  
## Fill out sections D, E, F, G, H, I, and J for adding a deployment  
##  
## Fill out section K for removing a deployment  
##  
##  
##  
#####  
##  
  
#####  
##  
#  
#  
#  
# SECTION A -
```

```
GENERAL                                     #
#
#
#####
##

#-----
--
# Specify the configuration option.
# Specify:
# - ADD      : for adding a new GoldenGate deployment.
# - REMOVE   : for removing an existing GoldenGate deployment.
#-----
--
CONFIGURATION_OPTION=ADD

#-----
--
# Specify the name for the new or existing deployment.
#-----
--
DEPLOYMENT_NAME=

#####
##
#
#
#                               SECTION B - ADMINISTRATOR
ACCOUNT                          #
#
#
# * If creating a new Service Manager, set the Administrator Account
username #
#    and
password.                                #
#
#
# * If reusing an existing Service
Manager:                              #
#    * Enter the credentials for the Administrator Account
in          #
#           the existing Service
Manager.                                  #
#
#
#####
##

#-----
--
# Specify the administrator account username for the Service Manager.
#-----
--
ADMINISTRATOR_USER=
```



```
#-----  
--  
# Specify the administrator account password for the Service Manager.  
#-----  
--  
ADMINISTRATOR_PASSWORD=  
  
#-----  
--  
# Optionally, specify a different administrator account username for the  
deployment,  
# or leave blanks to use the same Service Manager administrator credentials.  
#-----  
--  
DEPLOYMENT_ADMINISTRATOR_USER=  
  
#-----  
--  
# If creating a different administrator account username for the deployment,  
# specify the password for it.  
#-----  
--  
DEPLOYMENT_ADMINISTRATOR_PASSWORD=  
  
#####  
##  
#  
#  
#  
#          SECTION C - SERVICE  
MANAGER          #  
#  
#  
#####  
##  
  
#-----  
--  
# Specify the location for the Service Manager deployment.  
# This is only needed if the Service Manager deployment doesn't exist already.  
#-----  
--  
SERVICEMANAGER_DEPLOYMENT_HOME=  
  
#-----  
--  
# Optionally, specify a custom location for the Service Manager deployment  
ETC_HOME.  
#-----  
--  
SERVICEMANAGER_ETC_HOME=  
  
#-----  
--  
# Optionally, specify a custom location for the Service Manager deployment  
CONF HOME.
```

```

#-----
--
SERVICEMANAGER_CONF_HOME=

#-----
--
# Optionally, specify a custom location for the Service Manager deployment
SSL_HOME.
#-----
--
SERVICEMANAGER_SSL_HOME=

#-----
--
# Optionally, specify a custom location for the Service Manager deployment
VAR_HOME.
#-----
--
SERVICEMANAGER_VAR_HOME=

#-----
--
# Optionally, specify a custom location for the Service Manager deployment
DATA_HOME.
#-----
--
SERVICEMANAGER_DATA_HOME=

#-----
--
# Optionally, specify a custom location for the Service Manager deployment
ARCHIVE_HOME.
#-----
--
SERVICEMANAGER_ARCHIVE_HOME=

#-----
--
# Specify the host for the Service Manager.
#-----
--
HOST_SERVICEMANAGER=

#-----
--
# Specify the port for the Service Manager.
#-----
--
PORT_SERVICEMANAGER=

#-----
--
# Specify if SSL / TLS is or will be enabled for the deployment.
# Specify true if SSL / TLS is or will be enabled, false otherwise.
#-----
--

```

SECURITY_ENABLED=

```
#-----
--
# Specify if the deployment should enforce a strong password policy.
# Specify true to enable strong password policy management.
#-----
```

STRONG_PWD_POLICY_ENABLED=

```
#-----
--
# Specify if a new Service Manager should be created.
# Specify true if a new Service Manager should be created, false otherwise.
#
# This option is only needed when CONFIGURATION_OPTION is ADD.
#-----
```

CREATE_NEW_SERVICEMANAGER=

```
#-----
--
# Specify if Service Manager should be registered as a service/daemon. This
option is mutually exclusive with the 'INTEGRATE_SERVICEMANAGER_WITH_XAG'
option.
# Specify true if Service Manager should be registered as a service, false
otherwise.
#
# This option is only needed when CONFIGURATION_OPTION is ADD.
# This option does not apply to Windows platform.
#-----
```

REGISTER_SERVICEMANAGER_AS_A_SERVICE=

```
#-----
--
# Specify if Service Manager should be integrated with XAG. This option is
mutually exclusive with the 'REGISTER_SERVICEMANAGER_AS_A_SERVICE' option.
# Specify true if Service Manager should be integrated with XAG, false
otherwise.
#
# This option is only needed when CONFIGURATION_OPTION is ADD.
# This option is only supported for Oracle databases.
#-----
```

INTEGRATE_SERVICEMANAGER_WITH_XAG=

```
#-----
--
# If using an existing Service Manager, specify if it is integrated with XAG.
# Specify true if the existing Service Manager is integrated with XAG, false
otherwise.
#
# This option is only needed when CONFIGURATION_OPTION is ADD.
# This option is only supported for Oracle databases.
#-----
```

```
EXISTING_SERVICEMANAGER_IS_XAG_ENABLED=
#-----
--
# Specify if Remote Metrics using StatsD protocol will be enabled for the
Service Manager
# Specify true if Remote Metrics for the Service Manager will be enabled,
false otherwise
#-----
--
ENABLE_SERVICE_MANAGER_REMOTE_METRICS=
#-----
--
# If Remote Metrics for the Service Manager will be enabled, specify the
listening host
#-----
--
SERVICE_MANAGER_REMOTE_METRICS_LISTENING_HOST=
#-----
--
# If Remote Metrics for the Service Manager will be enabled, specify the
listening port for that server
#-----
--
SERVICE_MANAGER_REMOTE_METRICS_LISTENING_PORT=
#####
#
#
#
#           SECTION D - CONFIGURATION SERVICE
#
#
#####
#
#-----
--
# Specify if the Configuration Service will be enabled.
# Specify true if the Configuration Service will be enabled, false otherwise.
#-----
--
CONFIGURATION_SERVICE_ENABLED=
#-----
--
# Specify the Configuration Service backend type.
# Specify:
# - FILESYSTEM
# - ORACLE_DATABASE
#
# This is only needed if the Configuration Service will be enabled
#-----
```

```
CONFIGURATION_SERVICE_BACKEND_TYPE=ORACLE_DATABASE

#-----
--
# Specify the Configuration Service connection string for the database backend
#
# This is only needed if:
#     * The Configuration Service will be enabled
#     * CONFIGURATION_SERVICE_BACKEND_TYPE is ORACLE_DATABASE
#-----
--
CONFIGURATION_SERVICE_BACKEND_CONNECTION_STRING=

#-----
--
# Specify the Configuration Service username for the database backend
#
# This is only needed if:
#     * The Configuration Service will be enabled
#     * CONFIGURATION_SERVICE_BACKEND_TYPE is ORACLE_DATABASE
#-----
--
CONFIGURATION_SERVICE_BACKEND_USERNAME=

#-----
--
# Specify the Configuration Service password for the database backend
#
# This is only needed if:
#     * The Configuration Service will be enabled
#     * CONFIGURATION_SERVICE_BACKEND_TYPE is ORACLE_DATABASE
#-----
--
CONFIGURATION_SERVICE_BACKEND_PASSWORD=

#-----
--
# Specify the Configuration Service table name for the database backend
#
# This is only needed if:
#     * The Configuration Service will be enabled
#     * CONFIGURATION_SERVICE_BACKEND_TYPE is ORACLE_DATABASE
#-----
--
CONFIGURATION_SERVICE_BACKEND_TABLE_NAME=

#####
#
#
#
# SECTION E - SOFTWARE HOME
#
#
#
```

```
#####  
#  
  
#-----  
--  
# Specify the existing OGG software home location.  
#-----  
--  
OGG_SOFTWARE_HOME=  
  
#####  
#  
#  
#  
# SECTION F - DEPLOYMENT DIRECTORIES  
#  
#  
#  
#####  
#  
  
#-----  
--  
# Specify the location of the new or existing OGG deployment.  
#-----  
--  
OGG_DEPLOYMENT_HOME=  
  
#-----  
--  
# Specify the location for OGG_ETC_HOME.  
#-----  
--  
OGG_ETC_HOME=  
  
#-----  
--  
# Specify the location for OGG_CONF_HOME.  
#-----  
--  
OGG_CONF_HOME=  
  
#-----  
--  
# Specify the location for OGG_SSL_HOME.  
#-----  
--  
OGG_SSL_HOME=  
  
#-----  
--  
# Specify the location for OGG_VAR_HOME.  
#-----  
--  
OGG_VAR_HOME=
```

```
#-----
--
# Specify the location for OGG_DATA_HOME.
#-----
--
OGG_DATA_HOME=

#-----
--
# Specify the location for OGG_ARCHIVE_HOME.
#-----
--
OGG_ARCHIVE_HOME=

#####
#
#
#
#               SECTION G - ENVIRONMENT VARIABLES
#
#
#####
#

#-----
--
# Specify the value for the LD_LIBRARY_PATH environment variable.
#-----
--
ENV_LD_LIBRARY_PATH=

#-----
--
# Specify the value for the TNS_ADMIN environment variable.
# This environment variable is only for Oracle Databases.
#-----
--
ENV_TNS_ADMIN=

#-----
--
# This option is only needed when Sharding will be enabled.
# Specify the value for the STREAMS_POOL_SIZE environment variable.
# This environment variable is only for Oracle Databases.
#-----
--
ENV_STREAMS_POOL_SIZE=

#-----
--
# Specify any additional environment variables to be set in the deployment.
#-----
--
ENV_USER_VARS=
```

```
#####
#
#
#
#           SECTION H - SECURITY
#
#       This section is only needed if Security will be enabled
#
#
#####
#
#
#-----
# If security will be enabled, specify if TLS v1.2 will be enabled.
# Specify true if TLS v1.2 will be enabled, false otherwise.
#-----
--
--
TLS_1_2_ENABLED=

#
#-----
# If security will be enabled, specify if TLS v1.3 will be enabled.
# Specify true if TLS v1.3 will be enabled, false otherwise.
#-----
--
--
TLS_1_3_ENABLED=

#-----
--
# Specify if FIPS will be enabled.
#-----
--
FIPS_ENABLED=

#-----
--
# If SSL / TLS will be enabled, specify the server certificate
#-----
--
SERVER_CERTIFICATE=

#-----
--
# If importing a server certificate, specify the private key file in PKCS#8
format
# The private key file must not be encrypted
#-----
--
SERVER_CERTIFICATE_KEY_FILE=

#-----
--
```



```
# If importing a server certificate, optionally specify the CA certificates
file
#-----
--
SERVER_CA_CERTIFICATES_FILE=

#-----
--
# If SSL / TLS will be enabled, optionally specify the client certificate.
#-----
--
CLIENT_CERTIFICATE=

#-----
--
# If importing a client certificate, specify the private key file in PKCS#8
format
# The private key file must not be encrypted
#-----
--
CLIENT_CERTIFICATE_KEY_FILE=

#-----
--
# If importing a client certificate, optionally specify the CA certificates
file
#-----
--
CLIENT_CA_CERTIFICATES_FILE=

#####
#
#
#
#               SECTION I - SERVICES
#
#
#
#####
#
#-----
--
# Specify if the Administration server will be enabled.
# Specify true if the Administration server will be enabled, false otherwise.
#-----
--
ADMINISTRATION_SERVER_ENABLED=

#-----
--
# Required only if the Administration server will be enabled.
# Specify the port for Administration Server.
#-----
--
```

PORT_ADMINSRVR=

```
#-----
--
# Specify if the Distribution server will be enabled.
# Specify true if the Distribution server will be enabled, false otherwise.
#-----
```

DISTRIBUTION_SERVER_ENABLED=

```
#-----
--
# Required only if the Distribution server will be enabled.
# Specify the port for Distribution Server.
#-----
```

PORT_DISTSRVR=

```
#-----
--
# If security is disabled, specify if this non-secure deployment will be used
# to send trail data to a secure deployment.
#-----
```

NON_SECURE_DISTSRVR_CONNECTS_TO_SECURE_RCVRSRVR=

```
#-----
--
# Specify if the Receiver server will be enabled.
# Specify true if the Receiver server will be enabled, false otherwise.
#-----
```

RECEIVER_SERVER_ENABLED=

```
#-----
--
# Required only if the Receiver server will be enabled.
# Specify the port for Receiver Server.
#-----
```

PORT_RCVRSRVR=

```
#-----
--
# Specify if Performance Metrics server will be enabled.
# Specify true if Performance Metrics server will be enabled, false otherwise.
#-----
```

METRICS_SERVER_ENABLED=

```
#-----
--
# Specify if Performance Metrics server is a critical service.
# Specify true if Performance Metrics server is a critical service, false
otherwise.
#
# This is optional and only takes effect when Performance Metrics server will
```

```

be enabled.
# Also, this option should only be set when the Service Manager is integrated
with XAG.
# The default value is false.
#
# This option is only supported for Oracle databases.
#-----
--
METRICS_SERVER_IS_CRITICAL=

#-----
--
# Specify the port for Performance Metrics server (TCP).
#
# This option is only needed when Performance Metrics server will be enabled.
#-----
--
PORT_PMSRVR=

#-----
--
# Specify the DataStore type for Performance Metrics server.
# Valid values are: BDB, LMDB
#
# This option is only needed when Performance Metrics server will be enabled.
#-----
--
PMSRVR_DATASTORE_TYPE=

#-----
--
# Specify the DataStore home location for Performance Metrics server.
# This is optional and only takes effect when Performance Metrics server will
be enabled.
#-----
--
PMSRVR_DATASTORE_HOME=

#-----
--
# Specify if Remote Metrics using StatsD protocol will be enabled for the
Deployment
# Specify true if Remote Metrics for the deployment will be enabled, false
otherwise
#-----
--
ENABLE_DEPLOYMENT_REMOTE_METRICS=

#-----
--
# If Remote Metrics for the deployment will be enabled, specify the listening
host
#-----
--
DEPLOYMENT_REMOTE_METRICS_LISTENING_HOST=

```

```
#-----  
--  
# If Remote Metrics for the deployment will be enabled, specify the listening  
port for that server  
#-----  
--  
DEPLOYMENT_REMOTE_METRICS_LISTENING_PORT=  
  
#####  
#  
#  
#  
# SECTION J - REPLICATION OPTIONS  
#  
#  
#  
#####  
#  
#-----  
--  
# Specify the value for the GoldenGate schema.  
#-----  
--  
OGG_SCHEMA=  
  
#####  
#  
#  
#  
# SECTION K - REMOVE DEPLOYMENT OPTIONS  
#  
#  
#  
#####  
#  
#-----  
--  
# Specify if the deployment files should be removed from disk.  
# Specify true if the deployment files should be removed, false otherwise.  
#-----  
--  
REMOVE_DEPLOYMENT_FROM_DISK=
```

OGGCA Response File Example

You can use the following example OGGCA response file to see the type of values that can be entered when creating a deployment in silent mode.

Note

If you set `TLS_1_2_ENABLED= FALSE` but also provide the path for the `SERVER_CERTIFICATE` variable, the deployment creation will fail.

```
#####
## Copyright(c) Oracle Corporation 2016, 2026. All rights reserved.
##
##
##
## Specify values for the variables listed below to customize your
## installation.
##
##
## Each variable is associated with a comment. The comments can help to
## populate the variables with the appropriate values.
##
##
## IMPORTANT NOTE: This file should be secured to have read permission only
## by the Oracle user or an administrator who owns this configuration to
## protect any sensitive input values.
##
##
#####

#-----
--
# Do not change the following system generated value.
#-----
--
oracle.install.responseFileVersion=/oracle/install/
rspfmt_oggca_response_schema_v23_1_0

#####
##
##
##
## Oracle GoldenGate deployment configuration options and details
##
##
#####

#####
```

```
##
##
## Instructions to fill out this response file
##
## -----
##
## Fill out section A, B, and C for general deployment information
##
## Additionally:
##
## Fill out sections D, E, F, G, H, I, and J for adding a deployment
##
## Fill out section K for removing a deployment
##
##
##
#####
##

#####
##
#
#
# SECTION A -
GENERAL #
#
#
#####
##

#-----
--
# Specify the configuration option.
# Specify:
# - ADD : for adding a new GoldenGate deployment.
# - REMOVE : for removing an existing GoldenGate deployment.
#-----
--
CONFIGURATION_OPTION=ADD

#-----
--
# Specify the name for the new or existing deployment.
#-----
--
DEPLOYMENT_NAME=WEST_23ai

#####
##
#
#
# SECTION B - ADMINISTRATOR
ACCOUNT #
#
```

```
#
# * If creating a new Service Manager, set the Administrator Account
username #
# and
password. #
#
#
# * If reusing an existing Service
Manager: #
# * Enter the credentials for the Administrator Account
in #
# the existing Service
Manager. #
#
#
#####
##

#-----
--
# Specify the administrator account username for the Service Manager.
#-----
--
ADMINISTRATOR_USER=oggadmin

#-----
--
# Specify the administrator account password for the Service Manager.
#-----
--
ADMINISTRATOR_PASSWORD=Welcome##123

#-----
--
# Optionally, specify a different administrator account username for the
deployment,
# or leave blanks to use the same Service Manager administrator credentials.
#-----
--
DEPLOYMENT_ADMINISTRATOR_USER=oggadmin

#-----
--
# If creating a different administrator account username for the deployment,
# specify the password for it.
#-----
--
DEPLOYMENT_ADMINISTRATOR_PASSWORD=Welcome##123

#####
##
#
#
#
# SECTION C - SERVICE
MANAGER #
```

```

#
#
#####
##

#-----
--
# Specify the location for the Service Manager deployment.
# This is only needed if the Service Manager deployment doesn't exist already.
#-----
--
SERVICEMANAGER_DEPLOYMENT_HOME=/u02/deployments/srvmgr_26ai

#-----
--
# Optionally, specify a custom location for the Service Manager deployment
ETC_HOME.
#-----
--
SERVICEMANAGER_ETC_HOME=/u02/deployments/srvmgr_26ai/etc

#-----
--
# Optionally, specify a custom location for the Service Manager deployment
CONF_HOME.
#-----
--
SERVICEMANAGER_CONF_HOME=/u02/deployments/srvmgr_26ai/etc/conf

#-----
--
# Optionally, specify a custom location for the Service Manager deployment
SSL_HOME.
#-----
--
SERVICEMANAGER_SSL_HOME=/u02/deployments/srvmgr_26ai/etc/ssl

#-----
--
# Optionally, specify a custom location for the Service Manager deployment
VAR_HOME.
#-----
--
SERVICEMANAGER_VAR_HOME=/u02/deployments/srvmgr_26ai/var

#-----
--
# Optionally, specify a custom location for the Service Manager deployment
DATA_HOME.
#-----
--
SERVICEMANAGER_DATA_HOME=/u02/deployments/srvmgr_26ai/var/lib/data

#-----
--
# Optionally, specify a custom location for the Service Manager deployment

```



```

ARCHIVE_HOME.
#-----
--
SERVICEMANAGER_ARCHIVE_HOME=/u02/deployments/srvmgr_26ai/var/lib/archive

#-----
--
# Specify the host for the Service Manager.
#-----
--
HOST_SERVICEMANAGER=ogg_26ai

#-----
--
# Specify the port for the Service Manager.
#-----
--
PORT_SERVICEMANAGER=9011

#-----
--
# Specify if SSL / TLS is or will be enabled for the deployment.
# Specify true if SSL / TLS is or will be enabled, false otherwise.
#-----
--
SECURITY_ENABLED=false

#-----
--
# Specify if the deployment should enforce a strong password policy.
# Specify true to enable strong password policy management.
#-----
--
STRONG_PWD_POLICY_ENABLED=true

#-----
--
# Specify if a new Service Manager should be created.
# Specify true if a new Service Manager should be created, false otherwise.
#
# This option is only needed when CONFIGURATION_OPTION is ADD.
#-----
--
CREATE_NEW_SERVICEMANAGER=true

#-----
--
# Specify if Service Manager should be registered as a service/daemon. This
option is mutually exclusive with the 'INTEGRATE_SERVICEMANAGER_WITH_XAG'
option.
# Specify true if Service Manager should be registered as a service, false
otherwise.
#
# This option is only needed when CONFIGURATION_OPTION is ADD.
# This option does not apply to Windows platform.
#-----

```

```

--
REGISTER_SERVICEMANAGER_AS_A_SERVICE=false
#-----
--
# Specify if Service Manager should be integrated with XAG. This option is
mutually exclusive with the 'REGISTER_SERVICEMANAGER_AS_A_SERVICE' option.
# Specify true if Service Manager should be integrated with XAG, false
otherwise.
#
# This option is only needed when CONFIGURATION_OPTION is ADD.
# This option is only supported for Oracle databases.
#-----
--
INTEGRATE_SERVICEMANAGER_WITH_XAG=false

#-----
--
# If using an existing Service Manager, specify if it is integrated with XAG.
# Specify true if the existing Service Manager is integrated with XAG, false
otherwise.
#
# This option is only needed when CONFIGURATION_OPTION is ADD.
# This option is only supported for Oracle databases.
#-----
--
EXISTING_SERVICEMANAGER_IS_XAG_ENABLED=false

#-----
--
# Specify if Remote Metrics using StatsD protocol will be enabled for the
Service Manager
# Specify true if Remote Metrics for the Service Manager will be enabled,
false otherwise
#-----
--
ENABLE_SERVICE_MANAGER_REMOTE_METRICS=true

#-----
--
# If Remote Metrics for the Service Manager will be enabled, specify the
listening host
#-----
--
SERVICE_MANAGER_REMOTE_METRICS_LISTENING_HOST=localhost

#-----
--
# If Remote Metrics for the Service Manager will be enabled, specify the
listening port for that server
#-----
--
SERVICE_MANAGER_REMOTE_METRICS_LISTENING_PORT=8125

#####
#
#

```

```

#
#               SECTION D - CONFIGURATION SERVICE
#
#
#####
#

#-----
--
# Specify if the Configuration Service will be enabled.
# Specify true if the Configuration Service will be enabled, false otherwise.
#-----
--
CONFIGURATION_SERVICE_ENABLED=false

#-----
--
# Specify the Configuration Service backend type.
# Specify:
# - FILESYSTEM
# - ORACLE_DATABASE
#
# This is only needed if the Configuration Service will be enabled
#-----
--
CONFIGURATION_SERVICE_BACKEND_TYPE=FILESYSTEM

#-----
--
# Specify the Configuration Service connection string for the database backend
#
# This is only needed if:
#   * The Configuration Service will be enabled
#   * CONFIGURATION_SERVICE_BACKEND_TYPE is ORACLE_DATABASE
#-----
--
CONFIGURATION_SERVICE_BACKEND_CONNECTION_STRING=central.us.oracle.com:1521/
central_pdb1.us.oracle.com

#-----
--
# Specify the Configuration Service username for the database backend
#
# This is only needed if:
#   * The Configuration Service will be enabled
#   * CONFIGURATION_SERVICE_BACKEND_TYPE is ORACLE_DATABASE
#-----
--
CONFIGURATION_SERVICE_BACKEND_USERNAME=

#-----
--
# Specify the Configuration Service password for the database backend
#
# This is only needed if:

```

```

#      * The Configuration Service will be enabled
#      * CONFIGURATION_SERVICE_BACKEND_TYPE is ORACLE_DATABASE
#-----
--
CONFIGURATION_SERVICE_BACKEND_PASSWORD=

#-----
--
# Specify the Configuration Service table name for the database backend
#
# This is only needed if:
#      * The Configuration Service will be enabled
#      * CONFIGURATION_SERVICE_BACKEND_TYPE is ORACLE_DATABASE
#-----
--
CONFIGURATION_SERVICE_BACKEND_TABLE_NAME=

#####
#
#
#
#          SECTION E - SOFTWARE HOME
#
#
#
#####
#
#-----
--
# Specify the existing OGG software home location.
#-----
--
OGG_SOFTWARE_HOME=/u01/ogg

#####
#
#
#
#          SECTION F - DEPLOYMENT DIRECTORIES
#
#
#
#####
#
#-----
--
# Specify the location of the new or existing OGG deployment.
#-----
--
OGG_DEPLOYMENT_HOME=/u02/goldengate/deployments/WEST_26ai
#-----

```

```
# Specify the location for OGG_ETC_HOME.
#-----
--
OGG_ETC_HOME=/u02/deployments/WEST_26ai/etc

#-----
--
# Specify the location for OGG_CONF_HOME.
#-----
--
OGG_CONF_HOME=/u02/deployments/WEST_23ai/etc/conf

#-----
--
# Specify the location for OGG_SSL_HOME.
#-----
--
OGG_SSL_HOME=/u02/deployments/WEST_26ai/etc/ssl

#-----
--
# Specify the location for OGG_VAR_HOME.
#-----
--
OGG_VAR_HOME=/u02/deployments/WEST_26ai/var

#-----
--
# Specify the location for OGG_DATA_HOME.
#-----
--
OGG_DATA_HOME=/u02/deployments/WEST_26ai/var/lib/data

#-----
--
# Specify the location for OGG_ARCHIVE_HOME.
#-----
--
OGG_ARCHIVE_HOME=/u02/deployments/WEST_26ai/var/lib/archive

#####
#
#
#
#           SECTION G - ENVIRONMENT VARIABLES
#
#
#
#####
#

#-----
--
# Specify the value for the LD_LIBRARY_PATH environment variable.
#-----
```

```
ENV_LD_LIBRARY_PATH=${OGG_HOME}/lib/instantclient:${OGG_HOME}/lib

#-----
--
# Specify the value for the TNS_ADMIN environment variable.
# This environment variable is only for Oracle Databases.
#-----
--
ENV_TNS_ADMIN=/u02/deployments/WEST_23ai/etc

#-----
--
# This option is only needed when Sharding will be enabled.
# Specify the value for the STREAMS_POOL_SIZE environment variable.
# This environment variable is only for Oracle Databases.
#-----
--
ENV_STREAMS_POOL_SIZE=

#-----
--
# Specify any additional environment variables to be set in the deployment.
#-----
--
ENV_USER_VARS=

#####

#
#
#
#               SECTION H - SECURITY
#
#       This section is only needed if Security will be enabled
#
#
#
#####

#

#-----
# If security will be enabled, specify if TLS v1.2 will be enabled.
# Specify true if TLS v1.2 will be enabled, false otherwise.
#-----
--
TLS_1_2_ENABLED=false

#
#-----
# If security will be enabled, specify if TLS v1.3 will be enabled.
# Specify true if TLS v1.3 will be enabled, false otherwise.
#-----
--
TLS_1_3_ENABLED=false
```

```
#-----
--
# Specify if FIPS will be enabled.
#-----
--
FIPS_ENABLED=false

#-----
--
# If SSL / TLS will be enabled, specify the server certificate
#-----
--
#SERVER_CERTIFICATE=/home/oracle/OGG_Certs/Server_north_cert.pem
SERVER_CERTIFICATE=
#-----
--
# If importing a server certificate, specify the private key file in PKCS#8
format
# The private key file must not be encrypted
#-----
--
#SERVER_CERTIFICATE_KEY_FILE=/home/oracle/OGG_Certs/Server_north_key.pem
SERVER_CERTIFICATE_KEY_FILE=

#-----
--
# If importing a server certificate, optionally specify the CA certificates
file
#-----
--
#SERVER_CA_CERTIFICATES_FILE=/home/oracle/OGG_Certs/RootCA_cert.pem
SERVER_CA_CERTIFICATES_FILE=

#-----
--
# If SSL / TLS will be enabled, optionally specify the client certificate.
#-----
--
#CLIENT_CERTIFICATE=/home/oracle/OGG_Certs/DistClient_cert.pem
CLIENT_CERTIFICATE=

#-----
--
# If importing a client certificate, specify the private key file in PKCS#8
format
# The private key file must not be encrypted
#-----
--
#CLIENT_CERTIFICATE_KEY_FILE=/home/oracle/OGG_Certs/DistClient_key.pem
CLIENT_CERTIFICATE_KEY_FILE=

#-----
--
# If importing a client certificate, optionally specify the CA certificates
file
```

```

#-----
--
#CLIENT_CA_CERTIFICATES_FILE=/home/oracle/OGG_Certs/RootCA_cert.pem
CLIENT_CA_CERTIFICATES_FILE=

#####
#
#
#
#               SECTION I - SERVICES
#
#
#
#####
#

#-----
--
# Specify if the Administration server will be enabled.
# Specify true if the Administration server will be enabled, false otherwise.
#-----
--
ADMINISTRATION_SERVER_ENABLED=true

#-----
--
# Required only if the Administration server will be enabled.
# Specify the port for Administration Server.
#-----
--
PORT_ADMINSRVR=9012

#-----
--
# Specify if the Distribution server will be enabled.
# Specify true if the Distribution server will be enabled, false otherwise.
#-----
--
DISTRIBUTION_SERVER_ENABLED=true

#-----
--
# Required only if the Distribution server will be enabled.
# Specify the port for Distribution Server.
#-----
--
PORT_DISTSRVR=9013

#-----
--
# If security is disabled, specify if this non-secure deployment will be used
# to send trail data to a secure deployment.
#-----
--
NON_SECURE_DISTSRVR_CONNECTS_TO_SECURE_RCVRSRVR=false

```



```

#-----
--
# Specify if the Receiver server will be enabled.
# Specify true if the Receiver server will be enabled, false otherwise.
#-----
--
RECEIVER_SERVER_ENABLED=true

#-----
--
# Required only if the Receiver server will be enabled.
# Specify the port for Receiver Server.
#-----
--
PORT_RCVRSRVR=9014

#-----
--
# Specify if Performance Metrics server will be enabled.
# Specify true if Performance Metrics server will be enabled, false otherwise.
#-----
--
METRICS_SERVER_ENABLED=true
#-----
--
# Specify if Performance Metrics server is a critical service.
# Specify true if Performance Metrics server is a critical service, false
otherwise.
#
# This is optional and only takes effect when Performance Metrics server will
be enabled.
# Also, this option should only be set when the Service Manager is integrated
with XAG.
# The default value is false.
#
# This option is only supported for Oracle databases.
#-----
--
METRICS_SERVER_IS_CRITICAL=false

#-----
--
# Specify the port for Performance Metrics server (TCP).
#
# This option is only needed when Performance Metrics server will be enabled.
#-----
--
PORT_PMSRVR=9015

#-----
--
# Specify the DataStore type for Performance Metrics server.
# Valid values are: BDB, LMDB
#
# This option is only needed when Performance Metrics server will be enabled.

```

```
#-----
--
PMSRVR_DATASTORE_TYPE=BDB

#-----
--
# Specify the DataStore home location for Performance Metrics server.
# This is optional and only takes effect when Performance Metrics server will
be enabled.
#-----
--
PMSRVR_DATASTORE_HOME=

#-----
--
# Specify if Remote Metrics using StatsD protocol will be enabled for the
Deployment
# Specify true if Remote Metrics for the deployment will be enabled, false
otherwise
#-----
--
ENABLE_DEPLOYMENT_REMOTE_METRICS=true

#-----
--
# If Remote Metrics for the deployment will be enabled, specify the listening
host
#-----
--
DEPLOYMENT_REMOTE_METRICS_LISTENING_HOST=localhost

#-----
--
# If Remote Metrics for the deployment will be enabled, specify the listening
port for that server
#-----
--
DEPLOYMENT_REMOTE_METRICS_LISTENING_PORT=8125

#####
#
#
#
#           SECTION J - REPLICATION OPTIONS
#
#
#####
#

#-----
--
# Specify the value for the GoldenGate schema.
#-----
--
```

```
OGG_SCHEMA=oggadmin
```

```
#####
#
#
#
#          SECTION K - REMOVE DEPLOYMENT OPTIONS
#
#
#
#####
#

#-----
--
# Specify if the deployment files should be removed from disk.
# Specify true if the deployment files should be removed, false otherwise.
#-----
--
REMOVE_DEPLOYMENT_FROM_DISK=
```

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, RMTTRAIL, 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/tr0000000000
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)      FORMATTPE :      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)
RECLLENGTH   :     18 (X0012)        IO TIME      : 2019/01/17 16:47:58.000.000
IOTYPE       :      5 (X05)          ORIGNODE     :    255 (XFF)
TRANSIND     :      . (X00)          FORMATTYPER :      R (X52)
SYSKEYLEN    :      0 (X00)          INCOMPLETE  :      . (X00)
AUDITRBA     :      15              AUDITPOS    : 13287580
CONTINUED    :      N (X00)          RECCOUNT    :      1 (X01)

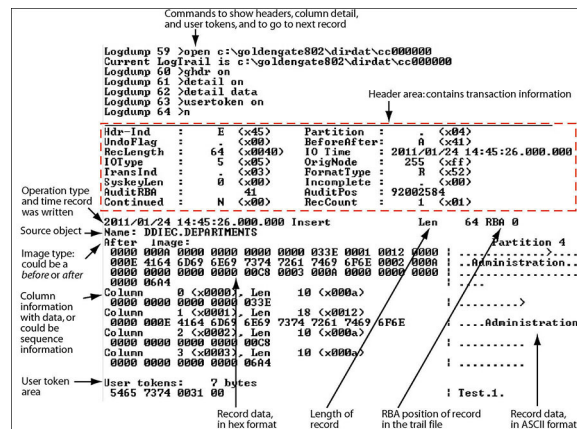
2019/01/17 16:47:58.000.000 INSERT          LEN      18 RBA 3486
NAME: TKGGU1.T1 (PARTITION: P1, TDR/PNR INDEX: 1/2)
AFTER IMAGE:                                PARTITION X8C      G      B
0000 0500 0000 0100 3101 0005 0000 0001 0031 | .....1.....1
COLUMN      0 (X0000), LEN      5 (X0005)
0000 0100 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.
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 UPDATERECORDFORMAT 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 UPDATERECORDFORMAT 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.

Field	Description
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.
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 when the change was committed to the source database
- 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 TKN-OPTYPE TKN-LENGTH TKN-TRAN_IND	: syshq : EXTORA : AFTER : 2011-01-24 17:08:59.000000 : 3604496 : 4058 : 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
3-Delete	A record/row was deleted. A Delete 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 Insert 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 Update 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 TMF <code>AuditComp</code> format was updated. In this format, only the changed bytes are present. A 4-byte descriptor in the format of <code>2-byte_offset2-byte_length</code> 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

Type	Description	Platform
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 FieldComp format.	All
16-FileRename	A file was renamed.	NSK
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

Type	Description	Platform
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
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 UNIFIED UPDATE contains all of the columns that are available in the transaction record for both the before and after images. The before image in a UNIFIED UPDATE 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 UPDATE.	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 BULKIO operation was performed. The Oracle GoldenGate record contains the RAW DP2 block.	NSK non-TMF

Type	Description	Platform
201 through 204	These are different types of NonStop trace records. Trace records are used by Oracle GoldenGate support analysts. The following are descriptions. - 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 203 — unused - ARTYPE_COLLECTORTS_GGS 204 — unused	NSK non-TMF
205-GGSCComment	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 SAVE command.	All

Type	Description	Platform
249 through 254	These are different types of NonStop trace records. Trace records are used by Oracle GoldenGate support analysts. The following are descriptions. - ARTYPE_LOGGER_ADDED_STATS 249 — 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) - ARTYPE_LIBRARY_OPEN 250 — written by BASELIB to show that the application opened a file - ARTYPE_LIBRARY_CLOSE 251 — written by BASELIB to show that the application closed a file. - ARTYPE_LOGGER_ADDED_OPEN 252 — unused - ARTYPE_LOGGER_ADDED_CLOSE 253 — unused	NSK non-TM

Type	Description	Platform
	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, Seqno 47, RBA 68748800
Log Read Checkpoint File /orarc/oradata/racq/redo04.log
  2011-01-01 14:16:19 Thread 2, Seqno 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
GGS 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/eh0000000000
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
```

Oracle GoldenGate Microservices Roles

This topic provides a comprehensive list of roles required for performing different tasks in Oracle GoldenGate Microservices Architecture.

The following table describes the activities that can be performed by each user role:

Table 13-1 Oracle GoldenGate User Roles and Privileges

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 server.
Administrator	Grants full access to the user, including the ability to alter general, non-security related operational parameters and profiles of the server.
Security	Grants administration of security related objects and invoke security related service requests. This role has full privileges.

Based on these user roles, you can perform the activities mentioned in the following table.

Table 13-2 Oracle GoldenGate Microservices Architecture Roles and Tasks

Admin Client	Role ID	REST API Endpoint	Description
ADD CHECKPOINTTABLE	Administrator	Manage Checkpoint Tables	Creates a checkpoint table in a database.
ADD CREDENTIALS	Security	Create User	Create User credentials for use by the Admin Client.
ADD DISTPATH	Administrator	Create a new Oracle GoldenGate Distribution Path	Creates a distribution path.
ADD ENCRYPTIONPROFILE	Administrator	Create Encryption Profile	Create an encryption profile for trail encryption/decryption.
ADD EXTRACT	Administrator	Create Extract	Creates an Extract group.
ADD EXTTRAIL	Administrator	Create Trail	Adds a local trail to the Oracle GoldenGate configuration.
ADD HEARTBEATTABLE	Administrator	Create Heartbeat Table	Add a heartbeat table to the database
ADD MASTERKEY	Administrator	Create Version	Adds a master encryption key to a master encryption key wallet.

Table 13-2 (Cont.) Oracle GoldenGate Microservices Architecture Roles and Tasks

Admin Client	Role ID	REST API Endpoint	Description
ADD PROCEDURETRANDATA	Administrator	Manage Procedural Supplemental Logging	Enables procedure-level supplemental logging.
ADD PROFILE	Administrator	Create Configuration Value	Create a profile for managed processes.
ADD RECVPATH	Administrator	Create a new Oracle GoldenGate Collector Path	Creates a target-initiated distribution path in Receiver Service.
ADD REPLICAT	Administrator	Create Replicat	Adds a Replicat group.
ADD SCHEMATRANDATA	Administrator	Manage Schema Supplemental Logging	Enables schema-level supplemental logging.
ADD TRANDATA	Administrator	Manage Table Supplemental Logging	Enables table-level supplemental logging.
ALTER CREDENTIALS	Administrator	Update User	Alter User credentials for use by the Admin Client.
ALTER CREDENTIALSTORE	Administrator	Replace Alias	Changes the contents of a credentials store.
ALTER DISTPATH	Administrator	Update an existing distribution path.	Changes the attributes of a distribution path.
ALTER ENCRYPTIONPROFILE	Administrator	Replace Encryption Profile	Alter an encryption profile for trail encryption/decryption.
ALTER EXTRACT	Administrator	Update Extract	Changes attributes of an Extract group.
ALTER EXTTRAIL	Administrator	Update Trail	Changes attributes of a local trail.
ALTER HEARTBEATTABLE	Administrator	Update Heartbeat Table	Alter a heartbeat table in the database
ALTER RECVPATH	Administrator	Update an existing Oracle GoldenGate Collector Path	Changes the attributes of a target-initiated distribution path in Receiver Service.
ALTER REPLICAT	Administrator	Update Replicat	Changes attributes of a Replicat group.
CLEANUP CHECKPOINTTABLE	User	Manage Checkpoint Tables	Removes checkpoint records that are no longer needed.
CLEANUP EXTRACT	User	Issue Command	Deletes run history for an Extract group.
CLEANUP REPLICAT	User	Issue Command	Deletes run history of a Replicat group.
CLEAR INSTANTIATION CSN	Administrator	Manage Instantiation CSN	Clear the instantiation CSN on a target table.
CONNECT	User	Validate	Connect to an Oracle GoldenGate Service Manager
DBLOGIN USERIDALIAS	User	Validate	Logs the session into a database so that other commands that affect the database can be issued.

Table 13-2 (Cont.) Oracle GoldenGate Microservices Architecture Roles and Tasks

Admin Client	Role ID	REST API Endpoint	Description
DELETE CHECKPOINTTABLE	Administrator	Manage Checkpoint Tables	Removes a checkpoint table from a database.
DELETE CREDENTIALS	Administrator	Delete User	Remove Oracle GoldenGate user login credentials from the Admin Client.
DELETE CREDENTIALSTORE	Administrator	Delete Alias	Deletes the wallet that serves as a credentials store.
DELETE DISTPATH	Administrator	Delete an existing Oracle GoldenGate Distribution Path	Deletes a distribution path.
DELETE ENCRYPTIONPROFILE	Administrator	Delete Encryption Profile	Remove an encryption profile.
DELETE EXTRACT	Administrator	Delete Extract	Deletes an Extract group.
DELETE EXTTRAIL	Administrator	Delete Trail	Removes a local trail from the Oracle GoldenGate configuration.
DELETE HEARTBEATENTRY	Administrator	Delete Process Heartbeat Records	Delete heartbeat table entries from the database
DELETE HEARTBEATTABLE	Administrator	Delete Heartbeat Table	Delete a heartbeat table from the database
DELETE MASTERKEY	Administrator	Delete Version	Marks a master encryption key for deletion.
DELETE PROCEDURETRANDATA	Administrator	Manage Procedural Supplemental Logging	Disables procedure-level supplemental logging.
DELETE PROFILE	Administrator	Manage Instantiation CSN	Remove a managed process profile.
DELETE RECVPATH	Administrator	Delete Configuration Value	Deletes a target-initiated distribution path in Receiver Service.
DELETE REPLICAT	Administrator	Delete Replicat	Deletes a Replicat group.
DELETE SCHEMATRANDATA	Administrator	Manage Schema Supplemental Logging	Disables schema-level supplemental logging.
DELETE TRANDATA	Administrator	Manage Table Supplemental Logging	Disables table-level supplemental logging.
DISABLE SERVICE	Administrator	Update Service Properties	Disable the specified Oracle GoldenGate services.
ENABLE SERVICE	Administrator	Update Service Properties	Enable the specified Oracle GoldenGate services.

Table 13-2 (Cont.) Oracle GoldenGate Microservices Architecture Roles and Tasks

Admin Client	Role ID	REST API Endpoint	Description
ENCRYPT PASSWORD	User	Encrypt Data	Encrypt a password that is used in an Oracle GoldenGate parameter file or command.
FLUSH SEQUENCE	User	Execute Command	Updates an Oracle sequence so that initial redo records are available at the time that Extract starts capturing transaction data.
HEALTH DEPLOYMENT	User	Service Health Summary	Display the health of the specified Oracle GoldenGate deployments.
INFO ALL	User	Retrieve Extract + Retrieve Replicat	Displays status and lag for all Oracle GoldenGate processes on a system.
INFO CHECKPOINTTABLE	User	Manage Checkpoint Tables	Returns information about one or more checkpoint tables.
INFO CREDENTIALSTORE	User	List Domain Aliases and List Domains	Returns information about a credentials store.
INFO DISTPATH	User	Retrieve an existing Oracle GoldenGate Distribution Path Information	Returns information about distribution path(s).
INFO ENCRYPTIONPROFILE	User	Retrieve Encryption Profile	Returns information about encryption profiles.
INFO ER	User	Retrieve Extract and Retrieve Replicat	Returns information about the specified wildcarded groups.
INFO EXTRACT	User	Issue Command	Returns information about an Extract group.
INFO EXTTRAIL	User	Retrieve Trail	Returns information about a local trail.
INFO HEARTBEATTABLE	User	Retrieve Heartbeat Table	Describe a heartbeat table in the database
INFO MASTERKEY	User	List Versions	Returns information about master encryption keys.
INFO PARAM	User	Retrieve Parameter Info	Displays parameter definition information.
INFO PROCEDURETRANDATA	User	Manage Procedural Supplemental Logging	Returns information about the state of procedure-level supplemental logging.
INFO PROFILE	User	Retrieve Configuration Value	Returns information about managed process profiles.

Table 13-2 (Cont.) Oracle GoldenGate Microservices Architecture Roles and Tasks

Admin Client	Role ID	REST API Endpoint	Description
INFO RECVPATH	User	Retrieve an existing Oracle GoldenGate Receiver Service Path Information	Returns information about target-initiated distribution path(s) in Receiver Service.
INFO REPLICAT	User	Issue Command	Returns information about a Replicat group.
INFO SCHEMATRANDATA	User	Manage Schema Supplemental Logging	Returns information about the state of schema-level supplemental logging.
INFO TRANDATA	User	Manage Table Supplemental Logging	Returns information about the state of table-level supplemental logging.
KILL ER	Operator	KILL EXTRACT + KILL REPLICAT	Forcibly terminate the specified wildcarded groups.
KILL EXTRACT	Operator	Issue Command	Forcibly terminates the run of an Extract group.
KILL REPLICAT	Operator	Update Replicat	Forcibly terminates a Replicat group.
LAG ER	User	LAG EXTRACT + LAG REPLICAT	Returns lag information about the specified wildcarded groups.
LAG EXTRACT	User	Issue Command	Returns information about Extract lag.
LAG REPLICAT	User	Issue Command	Returns information about Replicat lag.
LIST TABLES	User	Retrieve Database Tables	Lists the tables in the database with names that match the input specification.
MININGDBLOGIN USERIDALIAS	User	Validate	Specifies the credentials of the User that an Oracle GoldenGate process uses to log into an Oracle mining database.
PURGE EXTTRAIL	Administrator	Update Version	Removes files related to a local trail from the file system.
REGISTER EXTRACT	Administrator	Create Extract	Registers an Extract group with the database.
REGISTER REPLICAT	Administrator	Create Replicat	Registers a Replicat group with the database.
RENEW MASTERKEY	Administrator	Create Version	Adds a new version of a master encryption key.
RESTART DEPLOYMENT	Administrator	Update a Deployment	Restart the specified Oracle GoldenGate deployments.

Table 13-2 (Cont.) Oracle GoldenGate Microservices Architecture Roles and Tasks

Admin Client	Role ID	REST API Endpoint	Description
RESTART ER	Operator	Issue Command	Stops, then starts the specified wildcarded groups.
RESTART EXTRACT	Operator	Issue Command	Stops, then starts an Extract group.
RESTART REPLICAT	Operator	Issue Command	Stops, then starts a Replicat group.
RESTART SERVICE	Administrator	Update Service Properties	Restart the specified Oracle GoldenGate services.
SEND ER	User	SEND EXTRACT + SEND REPLICAT	Sends instructions to, or returns information about, the specified wildcarded groups.
SEND EXTRACT	User	Issue Command	Sends instructions to, or returns information about, a running Extract group.
SEND REPLICAT	User	Issue Command	Sends instructions to, or returns information about, a running Replicat group.
START DEPLOYMENT	Administrator	Update a Deployment	Start the specified Oracle GoldenGate deployments.
START DISTPATH	Operator	Update an existing distribution path.	Starts a distribution path.
START ER	Operator	Issue Command	Starts the specified wildcarded groups.
START EXTRACT	Operator	Issue Command	Starts an Extract group.
START RECVPATH	Operator	Update an existing Oracle GoldenGate Collector Path	Starts a target-initiated distribution path in Receiver Service.
START REPLICAT	Operator	Issue Command	Starts a Replicat group.
START SERVICE	Administrator	Update Service Properties	Start the specified Oracle GoldenGate services.
STATS DISTPATH	User	Retrieve an existing Oracle GoldenGate Distribution Path Statistics	Returns statistics for a distribution path.
STATS ER	User	Retrieve Report - Extract + Retrieve Report-Replicat	Returns processing statistics for the specified wildcarded groups.
STATS EXTRACT	User	Issue Command	Returns processing statistics for an Extract group.

Table 13-2 (Cont.) Oracle GoldenGate Microservices Architecture Roles and Tasks

Admin Client	Role ID	REST API Endpoint	Description
STATS RECVPATH	User	Retrieve an existing Oracle GoldenGate Receiver Service Path Stats	Returns statistics for a target-initiated distribution path in Receiver Service.
STATS REPLICAT	User	Issue Command	Returns processing statistics for a Replicat group.
STATUS DEPLOYMENT	User	Service Health Details	Display status of the specified Oracle GoldenGate deployments.
STATUS ER	User	STATUS EXTRACT + STATUS REPLICAT	Returns the state of the specified wildcarded groups.
STATUS EXTRACT	User	Issue Command	Returns the state of an Extract group.
STATUS REPLICAT	User	Retrieve Status	Returns the state of a Replicat group.
STATUS SERVICE	User	Retrieve Service	Display status of the specified Oracle GoldenGate services.
STOP DEPLOYMENT	Administrator	Update a Deployment	Stop the specified Oracle GoldenGate deployments.
STOP DISTPATH	Operator	Update an existing distribution path.	Stops a distribution path.
STOP ER	Operator	Issue Command	Stops the specified wildcarded groups.
STOP EXTRACT	Operator	Issue Command	Stops an Extract group.
STOP RECVPATH	Operator	Update an existing Oracle GoldenGate Collector Path	Stops a target-initiated distribution path in Receiver Service.
STOP REPLICAT	Operator	Issue Command	Stops a Replicat group.
STOP SERVICE	Administrator	Update Service Properties	Stop the specified Oracle GoldenGate services.
SYNCHRONIZE REPLICAT	Administrator	Issue Command	Returns all threads of a coordinated Replicat to a uniform start point after an unclean shutdown of the Replicat process.
UNDELETE MASTERKEY VERSION	Administrator	Update Version	Changes the state of a master encryption key from being marked as deleted to marked as available.
UNREGISTER EXTRACT	Administrator	(Part of Delete Extract)	Unregisters an Extract group from the database.

Table 13-2 (Cont.) Oracle GoldenGate Microservices Architecture Roles and Tasks

Admin Client	Role ID	REST API Endpoint	Description
UPGRADE CHECKPOINTTABLE	Administrator	Manage Checkpoint Tables	Adds a supplemental checkpoint table when upgrading Oracle GoldenGate from version 11.2.1.0.0 or earlier.
UPGRADE HEARTBEATTABLE	Administrator	Update Heartbeat Table	Alters heartbeat tables to track Extract restart position.
VALIDATE ENCRYPTIONPROFILE	Administrator	Validate Encryption Profile	Validates that data can be encrypted with the given profile.
VERSIONS	User	Execute Command	Displays information about the operating system and database.
VIEW DISCARD	User	Retrieve Report	Displays the discard file that is generated by Extract or Replicat.
VIEW ENCKEYS	User	Retrieve Configuration File	Retrieves the contents of a configuration file containing the ENCKEYS value.
VIEW GLOBALS	User	Retrieve Configuration File	Displays the contents of the GLOBALS parameter file in read-only mode on-screen.
VIEW MESSAGES	User	Retrieve Messages	Displays the Oracle GoldenGate message log (ggserr.log file).
VIEW PARAMS	User	Retrieve Configuration File	Displays the contents of a parameter file in read-only mode on-screen.
VIEW REPORT	User	Retrieve Report	Displays the process report that is generated by Extract or Replicat.

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 13-3 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 X BASIC 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
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

Table 13-3 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
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
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

Table 13-3 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
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
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

Table 13-3 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
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
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

Table 13-3 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
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
lv8pc8lr	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
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

Table 13-3 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
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
we8bs2000l5	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
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

Table 13-3 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
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
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

Identifier to use in parameter files and commands	Character set
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
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

Identifier to use in parameter files and commands	Character set
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
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

Identifier to use in parameter files and commands	Character set
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
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

Identifier to use in parameter files and commands	Character set
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
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

Identifier to use in parameter files and commands	Character set
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, Ukranian with Euro
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

Identifier to use in parameter files and commands	Character set
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
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

Identifier to use in parameter files and commands	Character set
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
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

Identifier to use in parameter files and commands	Character set
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
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_R0
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

Special Values Supported by Database Data Types

Type	Database	NaN	sNaN	Infinity/-Infinity
DECFLOAT	DB2 (All Platform)	Yes	Yes	Yes
BINARY FLOAT	Oracle, TimesTen	Yes	No	Yes
BINARY DOUBLE	Oracle, TimesTen	Yes	No	Yes
REAL	PostgreSQL, Teradata	Yes	No	Yes
DOUBLE	PostgreSQL, Teradata	Yes	No	Yes
NUMERIC	PostgreSQL	Yes	No	Yes
NUMBER	Oracle	No	No	Yes
NOT CURRENTLY SUPPORTED	Sybase, MySQL, MariaDB, Aurora, Azure, c-tree, MS SQL Server, Informix, HP SQL/MP	No	No	No

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.

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
RMTTRAIL 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.
Generated Source URI	Enter the location of the source trail file.
Target	Select ogg as the target protocol from the drop-down list. Enter the following in the given order: a. Target Hostname: Name of the target host service to which the connection will be established. b. Target Manager Port: Port number of the Oracle GoldenGate Classic Architecture Manager port. c. Target sub-directory for the trail file: Name of the subdirectory where the trail file is to be stored. For example, <code>.dirdat</code>. d. 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.

Options	Description
Target Encryption Algorithm	Select NONE from the drop-down list. To encrypt the target trail file, select the appropriate encryption algorithm from the drop-down list.
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	Select this option if you want the trail file in any of the following formats: • TEXT • SQL • XML
Encryption Profile	This is the encryption profile that was used to encrypt the trail file when it was generated. However, certain encryption methods are only available in Microservices Architecture and are not supported by Classic Architecture, so use this feature with caution.
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.