Oracle® Database

Oracle GoldenGate Microservices Documentation

(21c)

F70348-16
February 2025

Oracle Database Oracle GoldenGate Microservices Documentation, (21c)

F70348-16

Primary Author: Oracle Corporation

Contents

2 Install and Patch

3 Deploy

4 Prepare

7 Instantiate

8 Distribute

9 Replicat

10 Secure

11 Administer

12 Performance

13 Autonomous Database

14 Upgrade

15 Appendix

Preface

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

Audience

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

Documentation Accessibility

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

Access to Oracle Support

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

Related Information

The Oracle GoldenGate Product Documentation is available from the following location:

Oracle GoldenGate Documentation

Oracle GoldenGate for Big Data Documentation:

Oracle GoldenGate for Big Data

For OCI GoldenGate, refer to:

OCI GoldenGate

For details on Oracle Database High Availability, see:

Oracle Database High Availability

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, such as "From the File menu, select **Save**." Boldface also is used for terms defined in text or in the glossary.
italic *italic*	Italic type indicates placeholder variables for which you supply particular values, such as in the parameter statement: TABLE *table_name*. Italic type also is used for book titles and emphasis.
monospace MONOSPACE	Monospace type indicates code components such as user exits and scripts; the names of files and database objects; URL paths; and input and output text that appears on the screen. Uppercase monospace type is generally used to represent the names of Oracle GoldenGate parameters, commands, and user-configurable functions, as well as SQL commands and keywords.
UPPERCASE	Uppercase in the regular text font indicates the name of a 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: {*option1* | *option2* | *option3*}.
[ ]	Brackets within syntax indicate an optional element. For example in this syntax, the SAVE clause is optional: CLEANUP REPLICAT *group_name* [, SAVE *count*]. Multiple options within an optional element are separated by a pipe symbol, for example: [*option1* | *option2*].
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: extn, where ***n*** indicates 'north' compass direction. • Replicat: rep. Usage with location: repn, where ***n*** indicates 'north' compass direction. • Distribution Path: dp. Usage with location: dpn, where ***n*** indicates 'north' compass direction. • Checkpoint table: ggs_checkpointtable • Trail file names: e or d depending on whether the trail file is for the Extract of distribution path. Suffix derived in alphabetical order. Usage for an Extract trail file: ea, eb, ec. • Trail file subdirectory: The name will use compass directions to refer to the trail subdirectories. Example for trail subdirectory name would be /east, /west, /north, /south.

1

Concepts

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

Oracle GoldenGate

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

With Oracle GoldenGate, you can move committed transactions across multiple systems in your enterprise over a secure or non-secure configuration. It supports a wide range of databases and data sources, providing replication between same types or between heterogeneous databases. For example, you could replicate between an Oracle Autonomous Database instance and an Oracle Database instance, or between two Oracle Database instances set up as source and target, or a two-way replication between MySQL database and Oracle Database instances. In addition, you can replicate to Java Messaging Queues, flat files, and to Big Data in combination with Oracle GoldenGate for Big Data.

To know more, see https://www.oracle.com/middleware/technologies/goldengate.html.

Topics:

Why Do You Need Oracle GoldenGate?

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

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

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

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

When Do You Use Oracle GoldenGate?

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

Business Continuity and High Availability

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

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

Initial Load and Database Migration

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

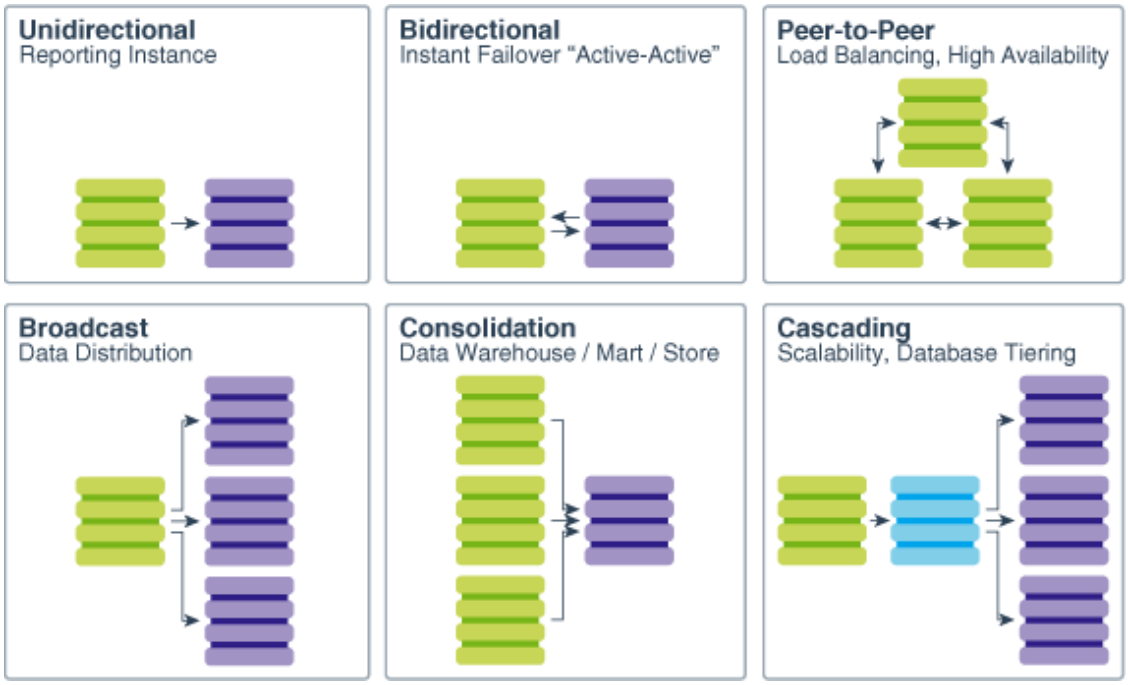
Data Integration

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

Topologies for Oracle GoldenGate

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

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



Oracle GoldenGate Product Family

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

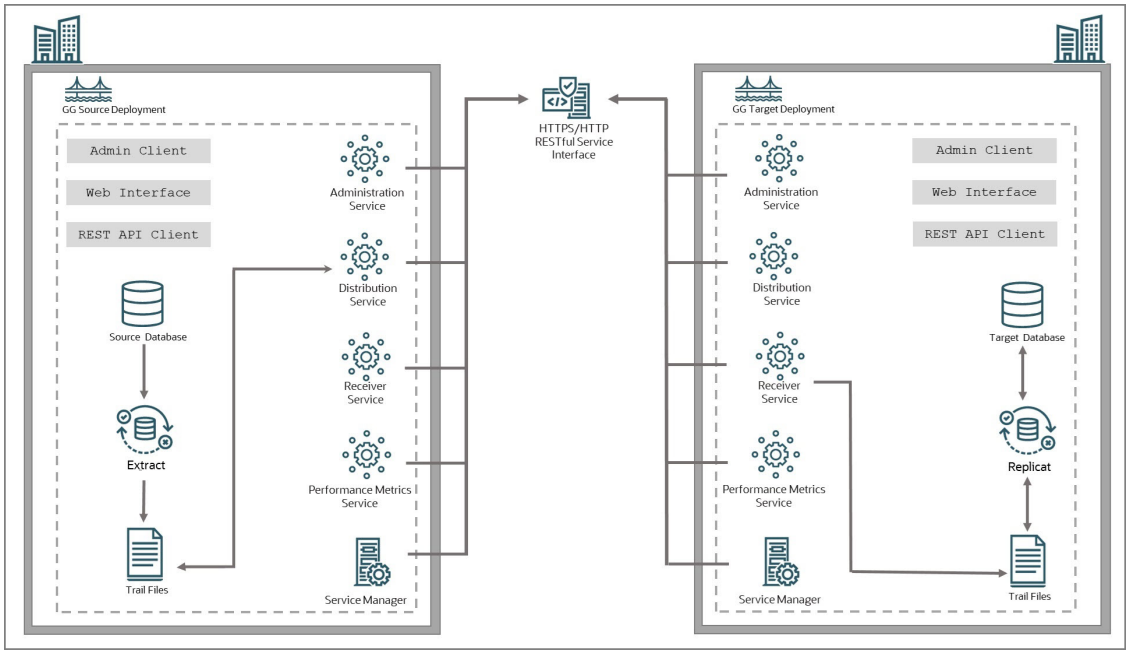
- **Oracle GoldenGate for Oracle Database**: Oracle GoldenGate Microservices for Oracle database provides all the data replication features of Oracle GoldenGate with the features of Oracle Database.
- **Oracle GoldenGate for Non-Oracle Databases**: Oracle GoldenGate Microservices for Oracle database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

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

Oracle GoldenGate Microservices Architecture

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

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



Topics:

Features of Oracle GoldenGate Microservices Architecture

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

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

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

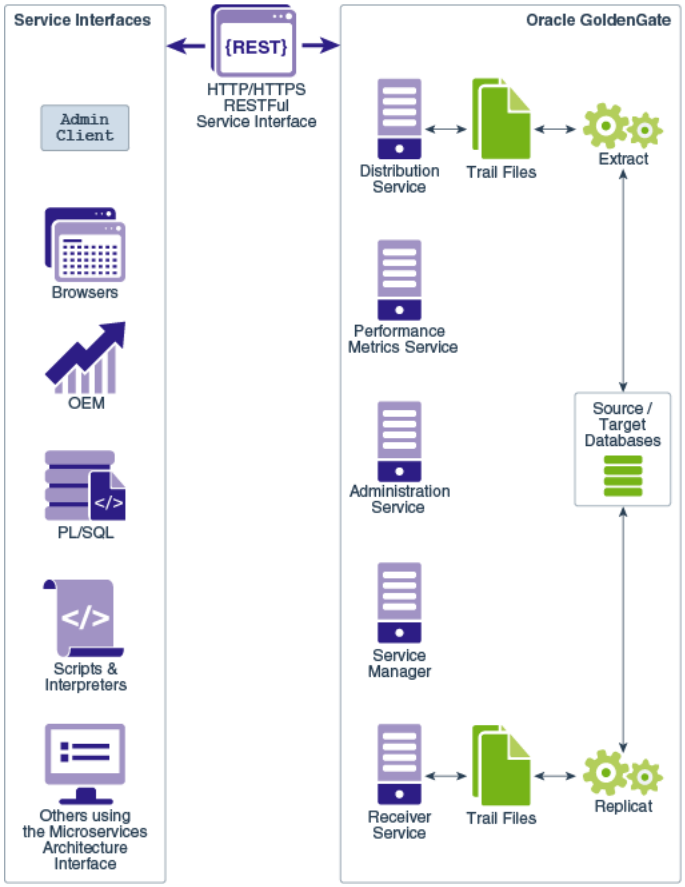
Access Points for Oracle GoldenGate Microservices

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

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

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

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



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

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

See About Extract and About Replicat.

The REST API provides service endpoints to manage Oracle GoldenGate deployments and replication services. See REST API Documentation.

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

Admin Client

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

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

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

REST API

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

See REST API Documentation.

Components of Oracle GoldenGate Microservices Architecture

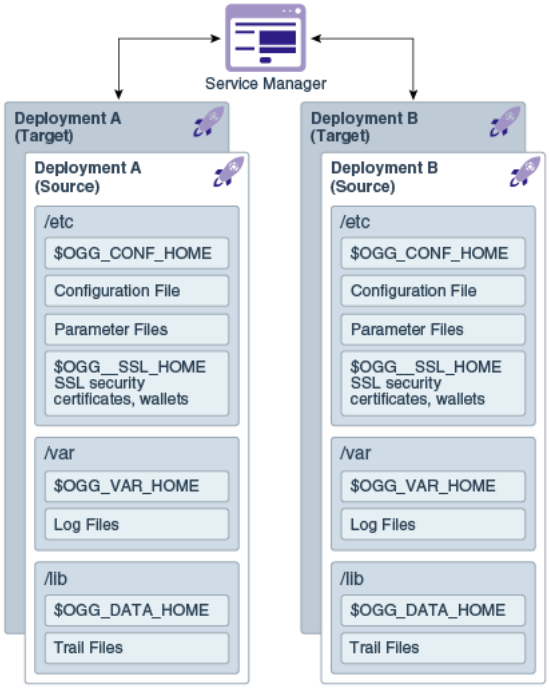
Directories and Variables in Microservices Architecture

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

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

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

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



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

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

Directory Name	Variable	Description	Default Directory Path
Deployment data home	OGG_DATA_HOME	The location where each deployment data artifacts (trail files) are stored.	/ogg_deployment_location/var/lib/data

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

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

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

Deployment

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. To learn more about these features of a deployment, see About Deployments.

Service Manager

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

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

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

See Manage Deployments from the Service Manager.

Administration Service

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

The Administration Service 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 Service 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 Service. See Admin Client for details.

The Administration Service 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 Service using REST JSON-RPC invocations that support REST API interfaces.

The Administration Service 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 Service 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

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:**
>
> A Distribution Service cannot filter data in the trail so it will send all operations.

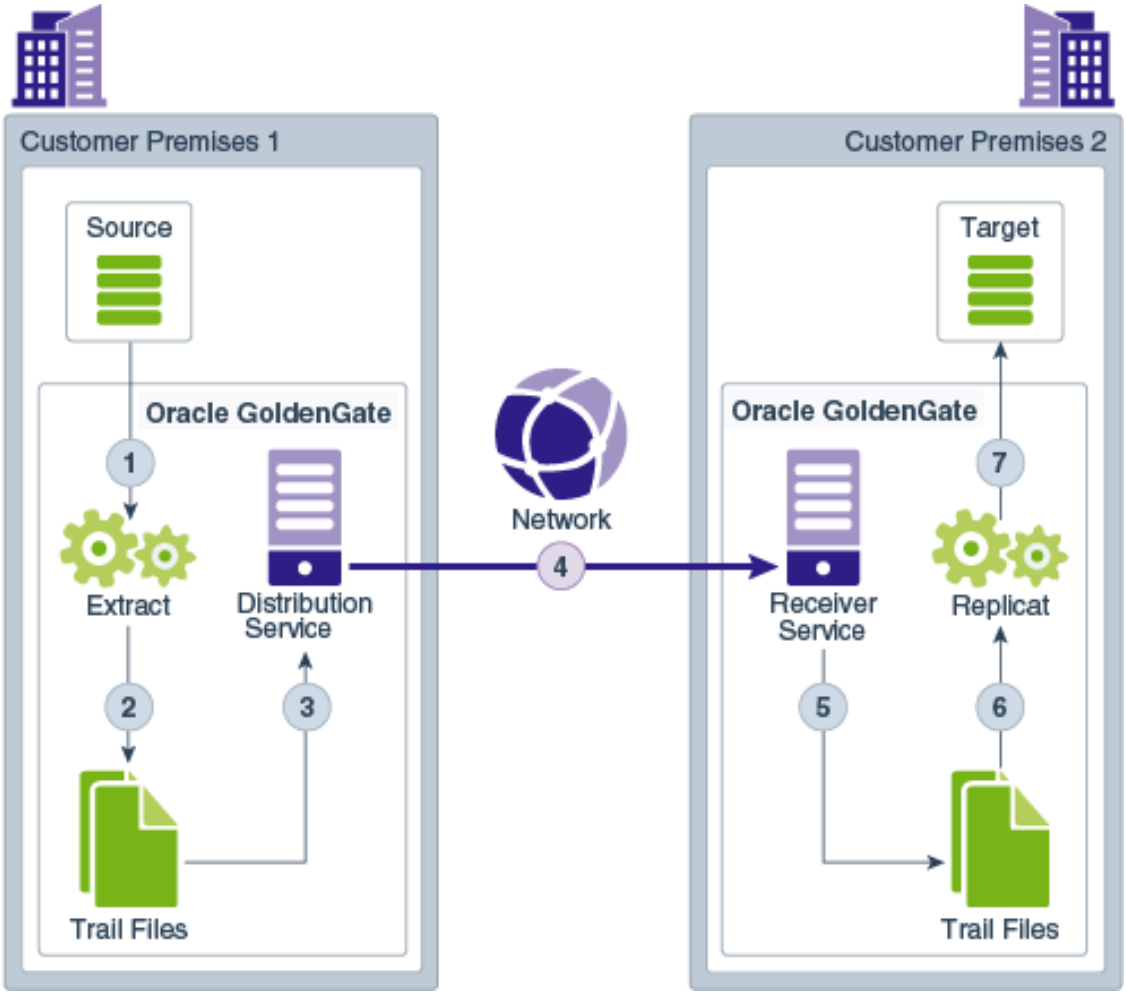
Receiver Service

A Receiver Service 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 Service 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 Service. It enables you to fully secure your data using SSL security. The Receiver Service seamlessly traverses through HTTP forward and reverse proxy servers.



Additionally, the Receiver Service 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 Service.

> **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.

Topics:

Target-Initiated Distribution Path

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

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

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

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

Components of Data Replication in Oracle GoldenGate

Types of Data Replication Configurations

Oracle GoldenGate can be configured for the following purposes:

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

Oracle GoldenGate Processes

Extract

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

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

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

See About Extract to learn more.

Replicat

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

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

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

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

- **Initial Loads**: When you set up Oracle GoldenGate for initial loads, the Replicat process applies a static data copy to target objects or routes the data to a high-speed bulk-load utility.
- **Change Synchronization**: When you set up Oracle GoldenGate to keep the target database synchronized with the source database, the Replicat process applies the source operations to the target objects using a native database interface or ODBC, depending on the database type.

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

See About Replicat to learn about different types of Replicats modes.

Distribution Paths for Data Transport

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

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

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.

Topics:

Processes that Write to the Trail File

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

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

Processes that Read from the Trail File

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

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

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

Trail File Creation and Maintenance

The trail files are created as needed during processing. You specify a two-character name for the trail when you add it to the Oracle GoldenGate configuration with the `ADD RMTTRAIL` or `ADD EXTTRAIL` command. By default, trails are stored in the dirdat sub-directory of the Oracle GoldenGate directory. You can specify a six or nine digit sequence number using the

`TRAIL_SEQLEN_9D | TRAIL_SEQLEN_6D GLOBALS` parameter; `TRAIL_SEQLEN_9D` is set by default. It is recommended to use the 9-digit sequence number when possible.

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

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

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

Parameter Files

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

Oracle GoldenGate Microservices Architecture uses the following runtime parameters:

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

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

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

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

See Working with Parameter Files to learn more.

Checkpoint Files

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

See Checkpoint Tables Additional Details .

2
Install and Patch

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

Download Oracle GoldenGate Software

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

Verify Certification and System Requirements

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

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

Here are some additional details about the supported platforms:

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

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

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

Operating System Requirements

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.

On a remote system

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**. Depending on the platform, the term **swap space** can be a swap partition, a swap file, or a shared memory segment (IBM i platforms).

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, 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 Reference for Oracle GoldenGate.

On the Db2 host system

Allocate approximately 10-50 MB of virtual memory for each Oracle GoldenGate log reader, `oggreadb`, that is invoked depending on the size of the log buffer. There is one invocation per Extract process on the remote system. To adjust the maximum log buffer size, use the `TRANLOGOPTIONS BUFSIZE` parameter in the Extract parameter file.

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. This is based on the IBM general guidelines available on IBM Support site.

Disk Requirements

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

Disk Requirements for Oracle GoldenGate Installation Files

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

Temporary Disk Requirements

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

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

> **Note:**
>
> `CACHEMGR` is an internally self-configuring and self-adjusting parameter. It is rare that this parameter requires modification. Doing so unnecessarily may result in performance degradation. It is best to acquire empirical evidence before opening an Oracle Service Request and consulting with Oracle Support.

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

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

Other Disk Space Considerations

In addition to the disk space required for the files and binaries that are installed by Oracle GoldenGate, allow additional disk space to hold the Oracle GoldenGate trails. Trails can be created up to 2GB in size, with a default of 500MB. The space required depends upon the

selected size of the trails, the amount of data being captured for replication, and how long the consumed trails are kept on the disk. The recommended minimum disk allocated for Trails may be computed as:

((transaction log size * 0.33) * number of log switches per day) * 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.

Db2 z/OS - Disk Requirements

On the Db2 host system

(Only applicable if you are installing stored procedures.) Assign a zFS (zSeries file systems) or hierarchical file system volume. To determine the size of the Oracle GoldenGate download file, examine the size of `zOSPrograms.zip` on the remote Db2 system after extracting the installation image.

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 Reverse Proxy with NGINX to Access Oracle GoldenGate Microservices.
- 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

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

- The user who installs Oracle GoldenGate must be granted read and write privileges on the Oracle GoldenGate software home directory.

- To install on Windows, the user who installs Oracle GoldenGate must log in as an Administrator.
- 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.
- Oracle recommends that you dedicate a database user to Oracle GoldenGate Extract and Replicat processes. Sensitive information might be available to anyone who runs an Oracle GoldenGate processes, depending on how database authentication is configured.
- For Db2 z/OS, the remote host requires privileges to use the `chmod +rw` command on the sub-directories in the Oracle GoldenGate product directory.

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 and the list of code page identifiers https://msdn.microsoft.com/en-us/library/windows/desktop/dd317756(v=vs.85).aspx.

Other Operating System Requirements

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

- To use Oracle GoldenGate user exits, install the C/C++ Compiler, which creates the programs in the required shared object or DLL.
- Gzip 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.

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:

- Db2 Connect v10.5 or greater
- IBM Data Server Driver for ODBC and CLI v10.5 or greater
- IBM Data Server Client v10.5 or greater
- IBM Data Server Runtime Client v10.5 or greater

Consider the following:

- Extract uses Open Database Connectivity (ODBC) to connect to 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 driver and is recommended for most configurations, although the other drivers are suitable also.
- 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

1. 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.
2. Install and run Db2 z/OS on the selected remote system so that the Replicat process can use the supplied DB2Connect driver.
3. 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
   ```

4. 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.

Oracle GoldenGate 21c for MySQL requires that OpenSSL 1.0 be installed on the Oracle GoldenGate server prior to creating a deployment.

Installing OpenSSL on Linux

OpenSSL 1.0 is included with the core operating system packages of OEL 7 and RHEL7 but is not included with OEL8/9 or RHEL 8/9, and therefore must be manually installed for these operating systems/versions.

To install the required OpenSSL 1.0 libraries (`libssl.so.10` and `libcrypto.so.10`), download and install the MySQL Connector/ODBC, version 8.0.17, using the following instructions:

1. Select version `8.0.17` 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 2.12) (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-8.0.17-linux-glibc2.12-x86-64bit
   ```

5. The OpenSSL 1.0 libraries needed for Oracle GoldenGate are located in the `mysql-connector-odbc-8.0.17-linux-glibc2.12-x86-64bit/lib` folder. This path needs to be added 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-8.0.17-linux-glibc2.12-x86-64bit/lib
   ```

Installing OpenSSL on Windows

OpenSSL 1.0 needs to be installed on the Windows server for Oracle GoldenGate for MySQL.

To install the required OpenSSL 1.0 libraries (`libcrypto-1_1-x64.dll` and `libssl-1_1-x64.dll`), download and install the MySQL Connector/ODBC, version 8.0.33, using the following instructions:

1. Select version **`8.0.33`** from the **Product Version** drop-down menu, using the link, https://downloads.mysql.com/archives/c-odbc/.
2. Select **Microsoft Windows** from the **Operating System** drop-down menu.
3. Click the **Download** link for the **`x86, 64-bit`** version of the package: ***Windows (x86, 64-bit), ZIP Archive***.
4. Save the file to a location of your choice, such as `C:\mysql` and extract the file, as shown in the following example:

   ```
   tar -xvf mysql-connector-odbc-noinstall-8.0.33-winx64.zip
   ```

5. The OpenSSL 1.0 libraries needed for Oracle GoldenGate are located in the `mysql-connector-odbc-noinstall-8.0.33-winx64\lib` folder. This path needs to be added to the `PATH` system variable, prior to creating a MySQL deployment, as shown in the following example.

   ```
   set PATH=%PATH%;C:\mysql\ mysql-connector-odbc-noinstall-8.0.33-winx64\lib
   ```

Prerequisites for Installing Oracle GoldenGate for Oracle Database

Oracle GoldenGate uses a single, unified build for capturing from and applying to multiple major Oracle Database versions for supported operating systems by including the latest Oracle database client libraries as part of Oracle GoldenGate. As a result, there are no prerequisites for installing Oracle GoldenGate for Oracle database.

Prerequisites to Configure Oracle GoldenGate Extract for PostgreSQL

To use Oracle GoldenGate 21.3 for PostgreSQL on OEL8, you must explicitly install OpenSSL 1.0 version on the OEL8 instance. This is necessary because the `libpq` library (packaged with Oracle GoldenGate 21c (21.3) for PostgreSQL), has a dependency on OpenSSL 1.0 version libraries, and these libraries are not available on OEL8 by default.

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
```

Prerequisites for Installing Oracle GoldenGate Microservices Architecture for SQL Server

Oracle GoldenGate for SQL Server must be installed on a supported operating system as per the Certification Matrix, and can be installed on the database server itself or on an application hub server, based on your preference.

Open a terminal session and using Microsoft's RedHat Enterprise Server installation instructions for adding the ODBC Drivers for Linux, perform the following steps with default values. Respond with 'y' when prompted.

```
sudo su #RedHat Enterprise Server 8
curl https://packages.microsoft.com/config/rhel/8/prod.repo >
/etc/yum.repos.d/mssql-release.repo
exit


sudo yum remove unixODBC-utf16 unixODBC-utf16-devel #to avoid conflicts
sudo ACCEPT_EULA=Y yum install msodbcsql17
sudo ACCEPT_EULA=Y yum install mssql-tools
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bash_profile
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bashrc
source ~/.bashrc
```

Additionally, observe the following program and settings for Oracle GoldenGate for SQL Server:

- Install either the Microsoft ODBC Driver 17 or Microsoft ODBC Driver 18 for the operating system where Oracle GoldenGate is to be installed:

  For Oracle GoldenGate on Windows, install the driver available at the following link:

  https://docs.microsoft.com/en-us/sql/connect/odbc/download-odbc-driver-for-sql-server?view=sql-server-2017

For Oracle GoldenGate on Linux, install the driver available at this link, and follow the instructions for RHEL and Oracle Linux packages:

https://learn.microsoft.com/en-us/sql/connect/odbc/linux-mac/installing-the-microsoft-odbc-driver-for-sql-server?view=sql-server-2017

> **Note:**
>
> Support for Microsoft ODBC Driver 18 was added with Oracle GoldenGate release version 21.8.0.0.3. Versions prior to release 21.8.0.0.3 do not support the Microsoft ODBC Driver 18 for SQL Server.

- To install capture on a remote Linux or Windows server, set the remote server's time and time zone to that of the database server, or use LSN based positioning for the Extract.

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:

http://support.oracle.com

Also see Installing Patches for Oracle GoldenGate Microservices Architecture.

Topics:

Installing Oracle GoldenGate Microservices Architecture

> **Note:**
>
> Oracle recommends using Oracle GoldenGate Microservices Architecture. From Oracle GoldenGate 21c onward, Oracle GoldenGate Classic Architecture for Oracle Database has been deprecated and may be desupported and unavailable in future releases.

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

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

- Operating System Requirements
- Prepare Databases

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

1. Install the Oracle GoldenGate software. See Performing an Interactive Installation with OUI for MA and Performing a Silent Installation with OUI.

2. Set the necessary environment variables for your database, if required.

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

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

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

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

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

Topics:

Performing an Interactive Installation with OUI for MA

Use the graphical user interface to install Oracle GoldenGate with prompts for required installation information. These instructions apply to new installations and upgrades:

1. Create a temporary staging directory where you will install Oracle GoldenGate.

   For example, on Linux the directory would be `mkdir /u01/stage/oggsc`.
   For Windows, create a directory where you will install Oracle GoldenGate such as, `gghome`, and a directory where you will keep the installation .zip file such as `ogg`. For example:
   `C:\>mkdir gghome ogg`

2. Extract the installation .zip file into the temporary staging directory. For example:

   For Linux: `unzip ./fbo_ggs_Linux_x64_services.zip -d ./`*`temp directory`*

   For Windows: `C:\ogg>unzip fbo_ggs_Windows_x64_Oracle_services_shiphome.zip -d temp`

3. From the expanded directory, run the `fbo_ggs_Linux_x64_Oracle_services_shiphome/Disk1/runInstaller` program on UNIX or Linux. On Windows, run the `fbo_ggs_Windows_x64_Oracle_services_shiphome/Disk1/setup.exe` program.

   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:
   a. For **Software Location**, specify the location where Oracle GoldenGate software will be installed. This will be your Oracle GoldenGate Home (`OGG_HOME`) after the installation is complete. For example: `C:\gghome` for Windows. 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.
   b. Click **Next**.
6. On the **Summary** page, confirm that there is enough space for the installation and that the installation selections are correct.
   a. (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.
   b. 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.
   c. 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.

You'll need to perform DataDirect driver installation for PostgreSQL after installing Oracle GoldenGate. See Install the DataDirect Driver for PostgreSQL for steps to perform this task.

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

You can edit the default response file without having to save the settings from an interactive Oracle Universal Installer session. If you are manually editing the response file, provide the following information and save the file:

- `INSTALL_OPTION` - The valid values are `DB2ZOS`, `MSSQL`, `MySQL`, `ORA21c`, and `PostgreSQL`. Set the value based on the database platform for the specific Oracle GoldenGate build to be installed.

  Example:

  ```
  INSTALL_OPTION=ORA21c
  ```

- `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/ogg21c_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
   ```

You'll need to perform DataDirect driver installation for PostgreSQL after installing Oracle GoldenGate. This is required only when you are installing Oracle GoldenGate for PostgreSQL. See Install the DataDirect Driver for PostgreSQL for steps to perform this task.

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.

Software Installation Directories and Programs for Oracle GoldenGate

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

Table 2-1 Directories in an Oracle GoldenGate MA installation

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

The following table describes the programs and utilities exclusively available with MA. Set the `$OGG_HOME/lib/instantclient` among other libraries that are used for database connectivity.

Name	Description	Default Directory
`adminsrvr`	The Administration Service supervises, administers, manages, and monitors processes operating within an Oracle GoldenGate deployment for both active and inactive processes.	`$OGG_HOME/bin`
`distsrvr`	A Distribution Service is a service that functions as a networked data distribution agent in support of conveying and processing data and commands in a distributed deployment.	`$OGG_HOME/bin`
`extract`	Extract data process.	`$OGG_HOME/bin`
`oggca.sh`	The MA Configuration Assistant.	`$OGG_HOME/bin`
`orapki`	Utility to manage public key infrastructure elements, such as wallets and certificate revocation lists.	`$OGG_HOME/bin`
`pmsrvr`	The Performance Metrics Service uses the metrics service to collect and store instance deployment performance results.	`$OGG_HOME/bin`
`recvsrvr`	A Receiver Service is the central control service that handles all incoming trail files.	`$OGG_HOME/bin`
`replicat`	Replicat data process.	`$OGG_HOME/bin`
`ServiceManager`	A Service Manager acts as a watchdog for other microservices in Oracle GoldenGate.	`$OGG_HOME/bin`

Name	Description	Default Directory
`sqlplus`	SQLPlus is an interactive, batch query tool installed with every Oracle Database Server or Client installation. It has a command-line user interface, a windows GUI, and a web-based user interface.	`$OGG_HOME/lib/instantclient`
`sql`	An SQL directory that contains the `healthcheck`, `legacy`, and `sharding` utilities.	`$OGG_HOME/lib`
`utl`	A utility directory that contains the `install`, `logging`, `reverseproxy`, and `sharding` utilities.	`$OGG_HOME/lib`

Post-installation Tasks

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

Install the DataDirect Driver for PostgreSQL

After installing Oracle GoldenGate for PostgreSQL, the Extract and Replicat processes use a DataDirect ODBC driver to connect to a PostgreSQL database. This driver is packaged with Oracle GoldenGate and needs to be installed and configured separately.

Installing for Linux
After installing Oracle GoldenGate for PostgreSQL on Linux, the DataDirect driver is *automatically installed*, but an `ODBCINI` variable needs to be set, if it has not already been done.

1. Create the **`ODBCINI`** variable for the Deployment from the deployment's **Configuration** tab of the **Service Manager Overview** page.
2. Set **Value** to the full path location of the `odbc` file where you will configure the database DSN connections. For example:

   Value equals: **`/etc/odbc.ini`**
3. Restart the deployment.

Installing Patches for Oracle GoldenGate Microservices Architecture

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

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

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

Topics:

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 `http://support.oracle.com`.
2. Log in with your Oracle ID and password.
3. Select the **Patches & Updates** tab.
4. On the **Search** tab, click **Product or Family**.
5. In the **Product** field, type **Oracle GoldenGate**.
6. From the **Release** drop-down list, select the patch version that you want to download.
7. Optionally, to limit the number of patches listed in the search results, select the required platform from the **Platform** drop-down list.
8. Click **Search**.
9. In the **Patch Advanced Search Results** list, select the patch that best meets your criteria.

   When you select a patch, a dialog box pops up under the build description, and then you are advanced to the patch details page.
10. Click the **Download** link for the patch and save the file to your system.

> **Note:**
>
> Before installing the patch, see Release Notes for Oracle Database for any new features, parameter changes, patching requirements, known issues, or bug fixes that affect your current configuration.

Patching Oracle GoldenGate Microservices Architecture Using OPatch

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

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

   Details from where to download OPatch are available at: How To Download And Install The Latest OPatch(6880880) Version (Doc ID 274526.1)
2. Download the Oracle GoldenGate patch and maintain a location for storing the contents of the patch ZIP file. This location or the absolute path is referred to as *`patch_top_dir`* in the subsequent steps.

3. Navigate to the *patch_top_dir* directory and run the following command to extract the contents of the patch ZIP file to the location you created previously.

   ```
   cd patch_top_dir
   ```

   ```
   unzip patch_number_version_platform.zip
   ```

4. Navigate to the unzipped patch directory:

   ```
   cd patch_top_dir/patch_number_dir
   ```

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

   For Linux: `$ export ORACLE_HOME=GoldenGate_Installation_Path`

   For Windows: `> set ORACLE_HOME=GoldenGate_Installation_Path`

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

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

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

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

   ```
   opatch lsinventory
   ```

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

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

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

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

   - Conflicts with a patch already applied to the `ORACLE_HOME`: In this case, stop the patch installation and contact Oracle Support Services.
   - Conflicts with a patch already applied to the `ORACLE_HOME` that is a subset of the patch you are trying to apply: In this case, continue with the patch installation because the new patch contains all the fixes from the existing patch in the `ORACLE_HOME`. The subset patch will automatically be rolled back prior to the installation of the new patch.

9. Before patching Oracle GoldenGate, if you have any deployments for the installation, ensure that you shut down all processes such as Extracts, Replicats, and Distribution paths, and stop all services for the deployments.

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

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

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

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

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

       ```
       OGG (not connected) 1>CONNECT https://host:srv_mgrport
       ```

```
DEPLOYMENT <deployment-name> AS <user> PASSWORD <password>
```

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

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

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

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

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

```
OGG (https://host:port deployment-name) exit

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

export OGG_VAR_HOME=OGG_SRVMGR_DIRECTORY/var

export OGG_ETC_HOME=OGG_SRVMGR_DIRECTORY/etc

OGG_SRVMGR_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. For more information, see Patching Oracle GoldenGate for PostgreSQL to Release Version 21.8.0.0.2 and Later.

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 for Microservices Architecture

This topic lists the post-patch installation tasks for non-Oracle databases, Microsoft SQL Server and MySQL.

Topics:

Patching Oracle GoldenGate MySQL 5.7 with DDL Replication Enabled

To patch Oracle GoldenGate MySQL 5.7 with DDL replication enabled:

1. Stop the metadata server using the following DDL install script `stop` option.

   ```
   ./ddl_install.sh stop user-id  password  port-number
   ```

2. Replace the `metadata_server` executable in the installation directory.
3. Start the metadata server running currently using ddl install script start option:

   ```
   ./ddl_install.sh start user-id  password  port-number
   ```

> **Note:**
>
> The DDL operations issued in between starting and stopping the `metadata_server` would be lost.

Patching Oracle GoldenGate for SQL Server - Extract Requirements

You must follow the existing patching procedures discussed in previous topics, Downloading Patches for Oracle GoldenGate and Patching Oracle GoldenGate Microservices Architecture Using OPatch. In addition, you must re-run `ADD TRANDATA` for each table that is already enabled for `TRANDATA` using these steps:

1. Stop all Oracle GoldenGate processes.
2. Follow normal patch procedures for binary replacement but do not start any Oracle GoldenGate processes. See Installing Patches for Oracle GoldenGate Microservices Architecture for details.
3. Manually stop the SQL Server CDC Capture job for the database. If the job is processing a large transaction, it may take some time before it actually stops.
4. Ensure that the Extract is stopped.
5. Using Admin client, run `ADD TRANDATA` again for every table that you previously enabled it for, including the heartbeat tables and any Replicat checkpoint table used as a `FILTERTABLE` object for active/active configurations.

   > **Note:**
   >
   > Do not run the `DELETE TRANDATA` command.

6. Manually restart the SQL Server CDC Capture job.
7. Manually restart the Oracle GoldenGate processes such as Extract, Replicat, and Manager.

Patching Oracle GoldenGate for PostgreSQL to Release Version 21.8.0.0.2 and Later

When patching Oracle GoldenGate for PostgreSQL from release versions prior to 21.8.0.0.2, to version 21.8.0.0.2 or later, you need to update the DSN entries in the `odbc.ini` file to take advantage of the new driver delivered as part of those patches.

Perform these steps after overwriting the existing version of Oracle GoldenGate for PostgreSQL and prior to restarting Oracle GoldenGate Extract and Replicat processes:

1. Update any existing DSN entries in the `odbc.ini` file and change the driver attributes for each DSN entry, to the following values:
   - **Driver** – For Oracle GoldenGate release versions 21.8.0.0.2 and later, set the value to: `/<GoldenGate_Installation_Folder>/datadirect/lib/ggpsql25.so`
2. Finish following the normal steps for patching Oracle GoldenGate.

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

To uninstall the patch, follow these steps:

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

   ```
   $ cd patch_top_dir/patch_number
   ```

3. Uninstall the patch by running the following command:

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

4. Start the services from the Oracle GoldenGate home.

Uninstalling Oracle GoldenGate Microservices Architecture

Learn about uninstallling 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.

Topics:

Removing Deployments and Service Manager

Learn how to remove a deployment using OGGCA.

Topics:

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. 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 **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.	• `/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`

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.

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
```

3
Deploy

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

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

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

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

About Deployments

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

What is a Deployment?

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

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

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

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

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

Secure Deployment

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

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

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

Non-Secure Deployment

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

Local and Remote Deployments

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

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.

Using OGGCA Wizard for Deployment

This section discusses using the OGGCA wizard for deployment.

Start the OGGCA Wizard

Adding deployments is the first task in the process of setting up a data replication platform. Deployments are managed from the Service Manager.

After completing the Oracle GoldenGate 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.

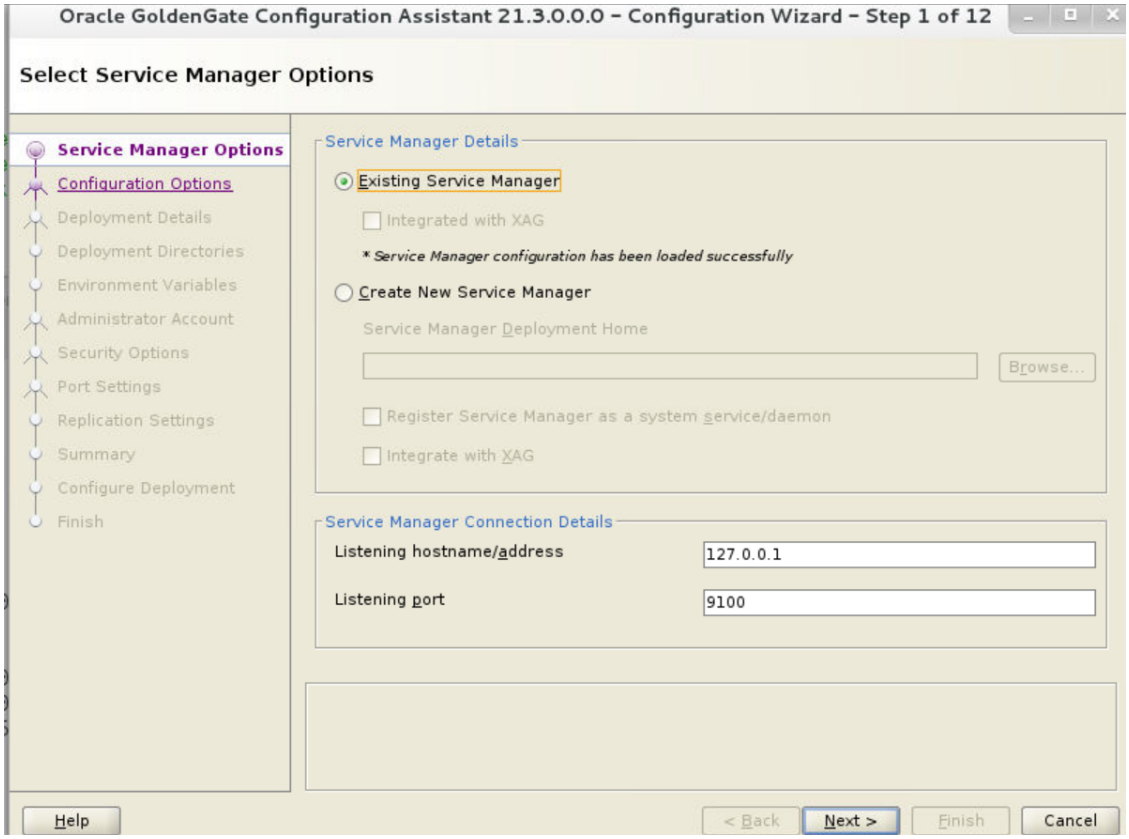
   or,

   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.

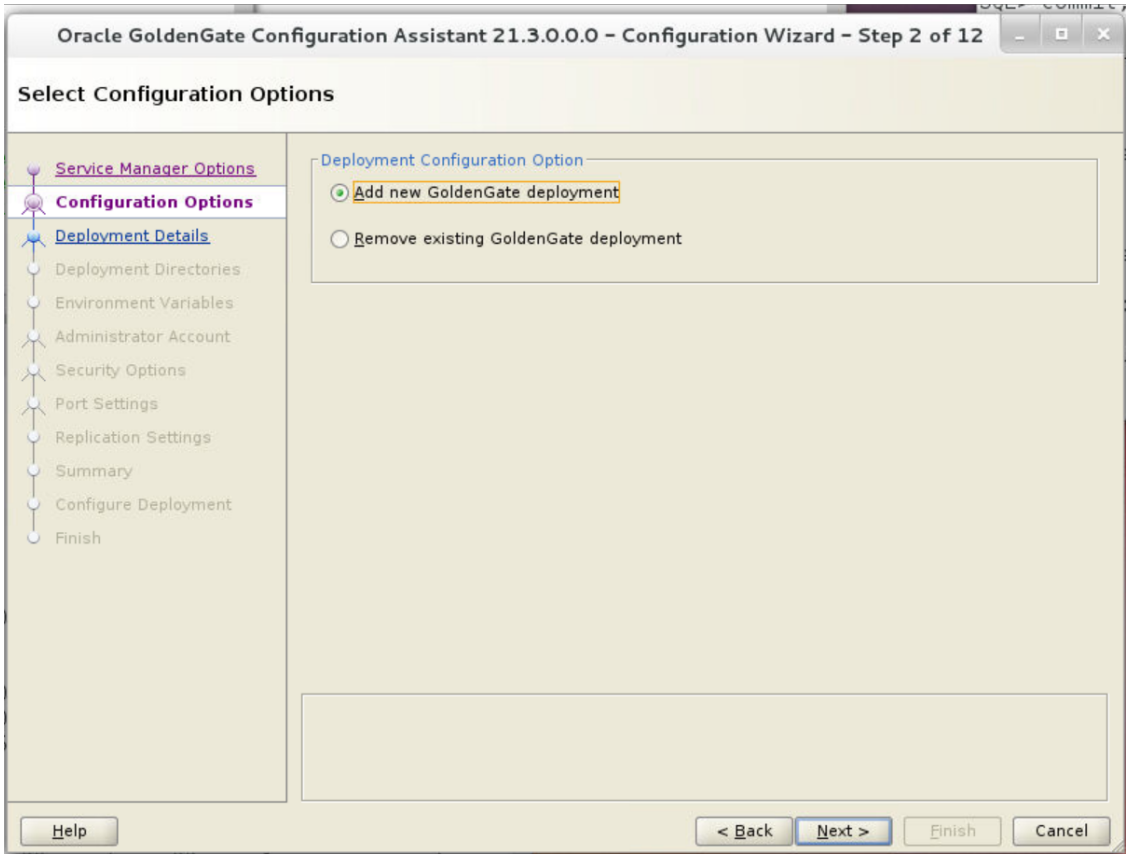
Select Service Manager Options



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 its 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.
2. For a new Service Manager, enter the **Service Manager Deployment Home** directory. Oracle recommends that you create a `ServiceManager` directory within the deployment sub-directory structure to store the Service Manager files.
3. Enter the connection details for the Service Manager.
   a. **Listening hostname/address**: Enter a hostname such as localhost or the IP address of the server where Service Manager will run.
   b. **Listening Port**: Enter a unique port number that the Service Manager will listen on, or choose the port already in use if selecting an existing Service Manager.
4. (Optional) Select the option **Register the Service Manager as a system service (daemon)** to avoid manually starting and stopping it if the machine is rebooted.

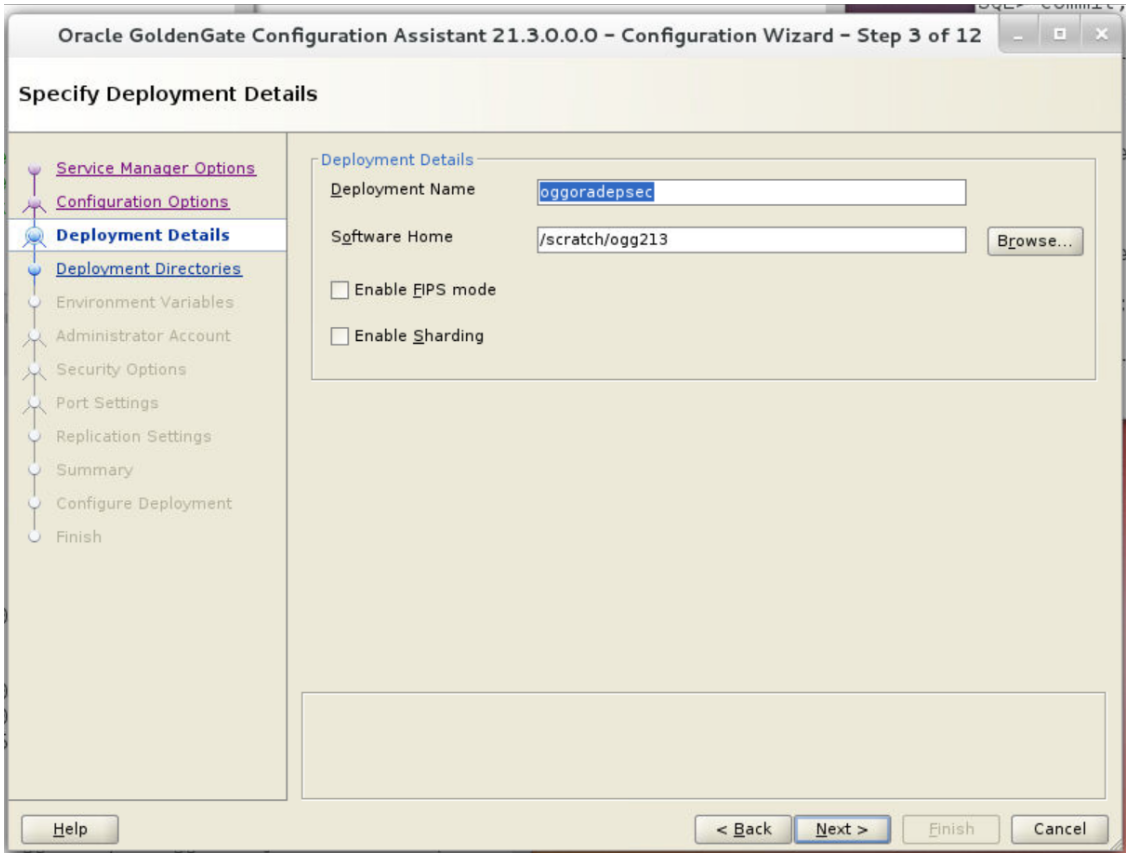
   If there is an existing Service Manager registered as a service and you select a new Service Manager to register as a service, an alert is displayed indicating that you cannot register the new one as a system service. All other Service Managers are started and stopped using scripts installed in the bin directory of the deployment.
5. (Optional) Select the **Integrate with XAG** option to integrate your deployment with an Oracle Grid Infrastructure for Oracle Database. This is only available for Oracle database in a cluster environment. This option cannot be used when running your Service Manager as a system service.
6. Click **Next**.

Configuration Options



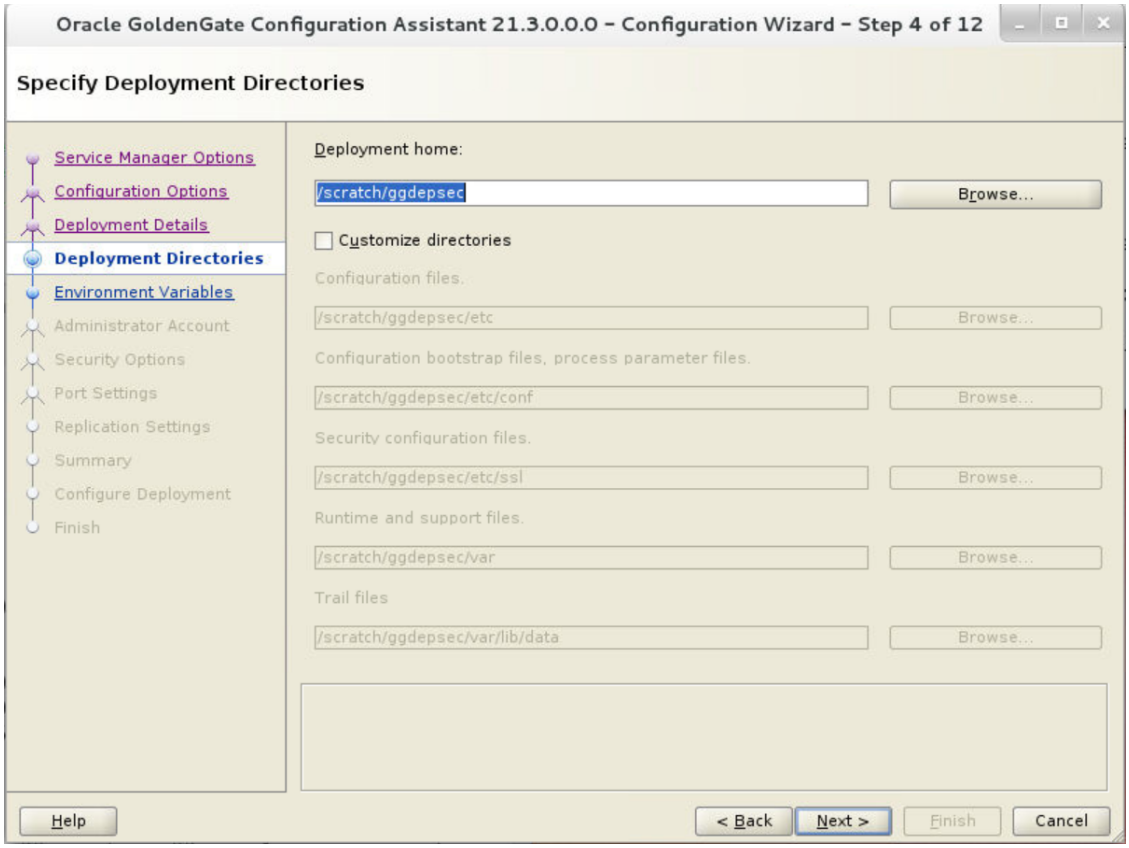
1. Click the **Add new GoldenGate deployment** option. You can only add or remove one deployment for one Service Manager at a time.
2. Click **Next**.

Deployment Details



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

Select Deployment Directories

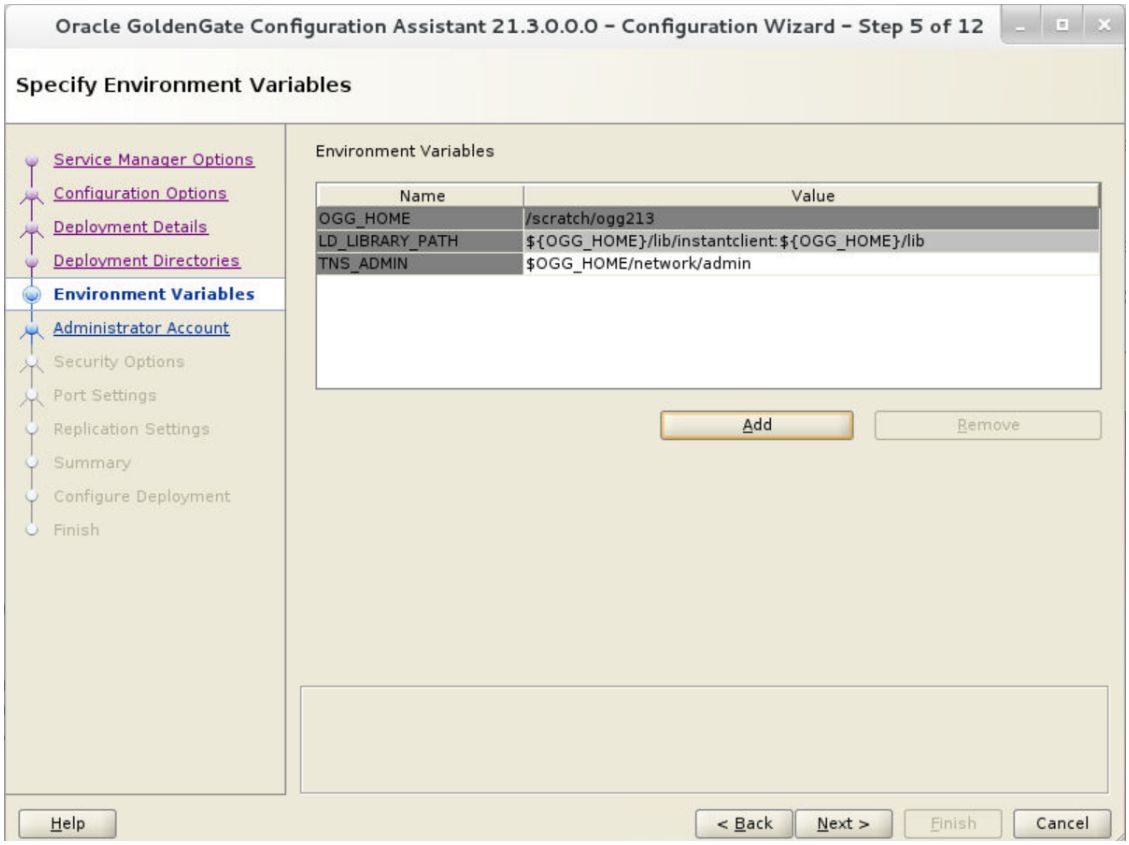


1. In the **Deployment home** box, specify a deployment directory to store the deployment registry and configuration files. Oracle recommends that you create a separate directory outside of the `$OGG_HOME` (installation directory) for easier upgrades. The additional fields are automatically populated based on the specified deployment directory.

> **Note:**
>
> The deployment directory name (user deployment directory) needs to be different than the directory name chosen in the first screen (Service Manager deployment directory). You can customize the deployment directories so that they are named and located differently from the default. Enter or select different directories for the various deployment elements. For deployment directory structure, see Directories and Variables in Microservices Architecture.

2. (Optional) Select the **Customize directories** check box, if you want to change the default locations for the Oracle GoldenGate configuration files.
3. Click **Next**.

Specify Environment Variables



1. Specify the values for the environment variables depending on database configurations. Double-click in the field to add or edit it. If you have previously set any of these environment variables, the value is automatically detected and populated in the respective environment variable field.

   OGG_HOME

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

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

   IBMCLIDRIVER

   Valid for Db2 z/OS.

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

   LD_LIBRARY_PATH

   This variable is used to specify the path to search for libraries on UNIX and Linux. It may have a different name on some operating systems, such as `LIBPATH` on IBM AIX on POWER Systems (64-Bit), and `SHLIB_PATH` on HP-UX. This path points to the Oracle GoldenGate installation directory and the underlying instant client directory by default.

If you are using User Exits, then append the `LD_LIBRARY_PATH` variable with the path to the additional shared libraries of the User Exit.

TNS_ADMIN

Valid for Oracle database.

This variable points to the directory location containing `tnsnames.ora`, which has the database connection details. This variable is optional.

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

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

STREAMS_POOL_SIZE

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

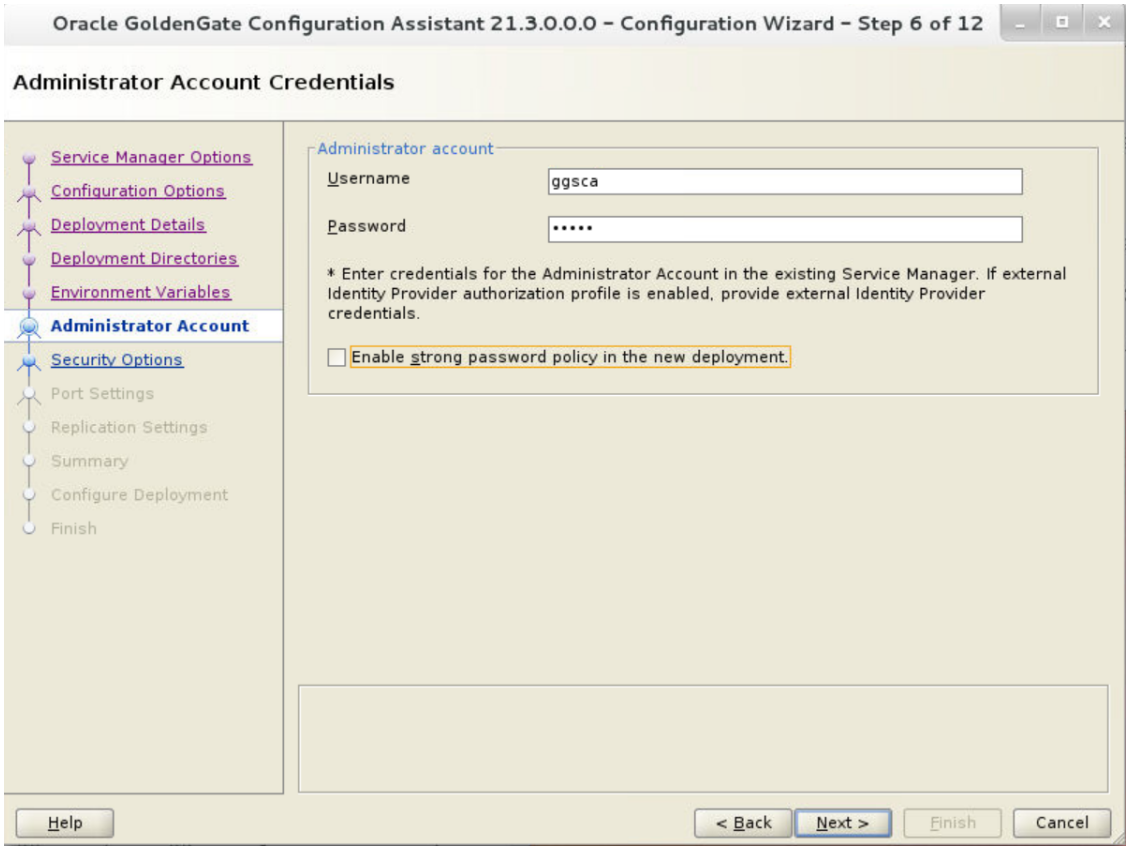
JAVA_HOME

If this variable is present during deployment creation, it will automatically be populated.

You can add or remove other environment variables to customize your deployment according to the database host.

2. Click **Next**.

Administrator Account



Configuration in this screen allows you to create credentials for the security user for Oracle GoldenGate. If this is not the first run of OGGCA, then you need to enter the administrator account credentials that are used to log in to the Service Manager because the deployment is getting added to this Service Manager.

If you want to integrate with Oracle Identity Cloud Service (IDCS) for authentication and authorization of users, then use this screen to specify the credentials for the IDCS account.

1. Enter a user name and password to log in to Oracle GoldenGate MA. This user is the security user for this deployment.
2. If you are using IDCS (as your external Identity Provider), then specify the user credentials for the IDCS server. On your first log in to the Service Manager, you need to enable the Authorization Profile for the Service Manager deployment. See Delegate User Authentication and Authorization to an External ID Provider.
3. 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 the Oracle GoldenGate security user only but not for IDCS default settings. See Manage Oracle Identity Cloud Service Password Policies section in *Administering Oracle Identity Cloud Service* guide for IDCS accounts.

> **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.

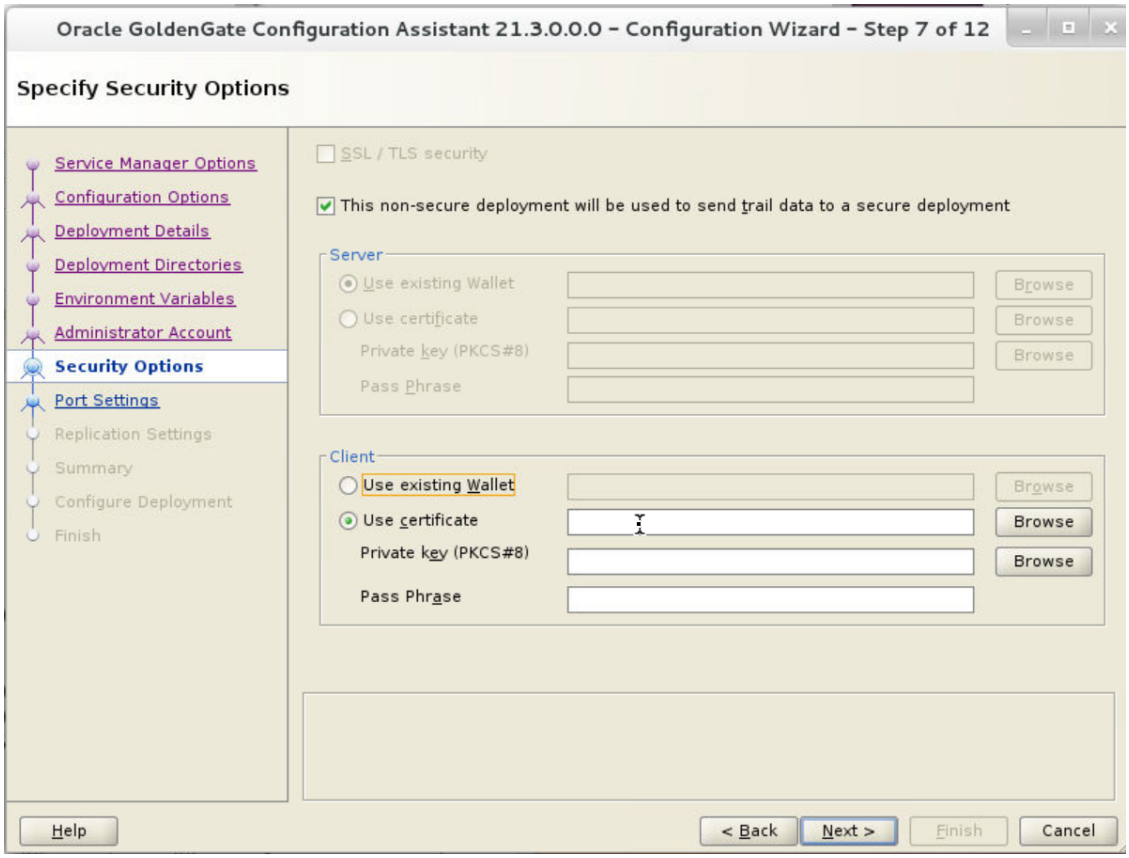
The strong password policy has the following requirements:

- At least one lowercase character [a...z]
- At least one upposercase 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 Add New Users to the Deployment.

1. If you are using an existing Service Manager, you must enter the same log in credentials that were created during the first run of OGGCA.
2. Select the **Enable a strong password policy** check box for the 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**.

Specify Security Options



1. Select the **SSL/TLS** security check box to enable security for the deployment. If you enabled Sharding for Oracle database, then must enable this option.
2. Deselect this option if you don't want to set up a secure deployment or want to use other types of security configurations such NGINX or reverse proxy support.
3. When you deselect the SSL/TLS security check box, the option **This non-secure deployment will be used to send trail data to a secure deployment** remains enabled. Select this check box to set up the deployment as a non-secure deployment that would send trail data to a secure target deployment.
4. In the **Server** (wallet or certificate) section, select one of the options, and then provide the required file locations. If you select the **Use existing wallet** option, the wallet directory must have the appropriate certificates already imported into it. If you choose to use a certificate, enter the corresponding pass phrase.

   When using a self-signed certificate, a new Oracle Wallet is created in the new deployment and these certificates are imported into it. For certificates, enter the location of the private key file and the pass phrase. The private key files must be in the `PKCS#8` format.
5. (Optional) The **Client** section is enabled if you select the **This non-secure deployment will be used to send trail data to a secure deployment** option. This option is useful when Distribution Service from the source deployment is not secure whereas the Receiver Service on the target deployment is secured. The sender (source) may be configured for public access while the Receiver Service requires authentication and authorization, which is established using PKI before the incoming data is applied. This option allows sending trail data to a secure deployment for environments such as DMZ where:

   If you select the **Use Existing Wallet** option, then specify the location of the existing wallet that stores the client certificates. Make sure that the certificates are already imported in the wallet directory.

If you select the **Use certificates** option, then enter the passphrase.

For more information, see Create Different Types of Certificates for a Secure Deployment.

Advanced Security Settings

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

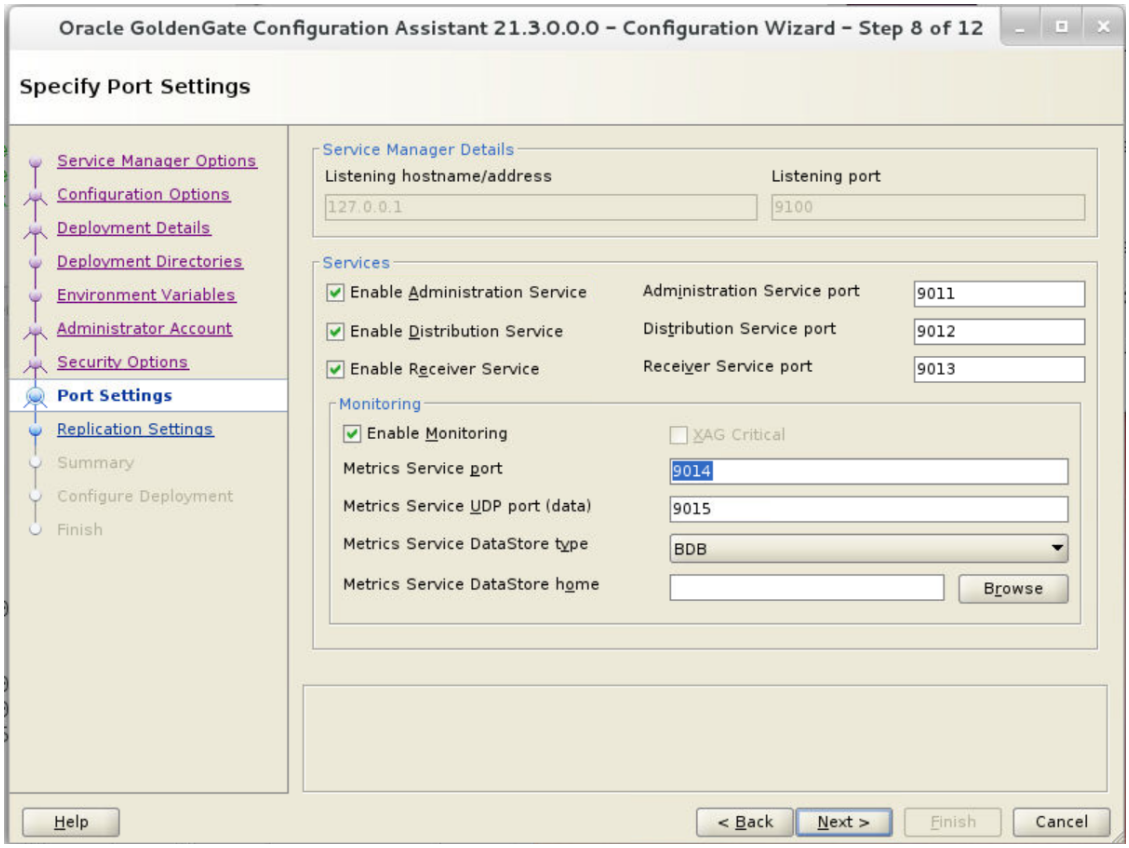
1. Use the arrows to add or remove cipher suites.
2. Use Up and Down to reorder how the cipher suites are applied and click **Next**.

Sharding Options

If Sharding is enabled on the Deployment Details screen, then this screen is displayed. You can specify the sharding options on this screen:

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

Port Settings



1. Enter the **Administration Service port** number, and then when you leave the field the other port numbers are populated in ascending numbers. Optionally, you can enter unique ports for each of the services.
2. Select **Enable Monitoring** to use the Performance Metrics Service.

> **Note:**
>
> For Oracle GoldenGate Microservices, selecting Enable Monitoring does not require Oracle GoldenGate Management Pack License. The license is required only when Enterprise Manager Plugin for GoldenGate is used to monitor Oracle GoldenGate Microservices instance.

3. Click inside the Performance Metrics Service port fields to populate or enter the ports you want to use. Ensure that you choose available ports for TCP.

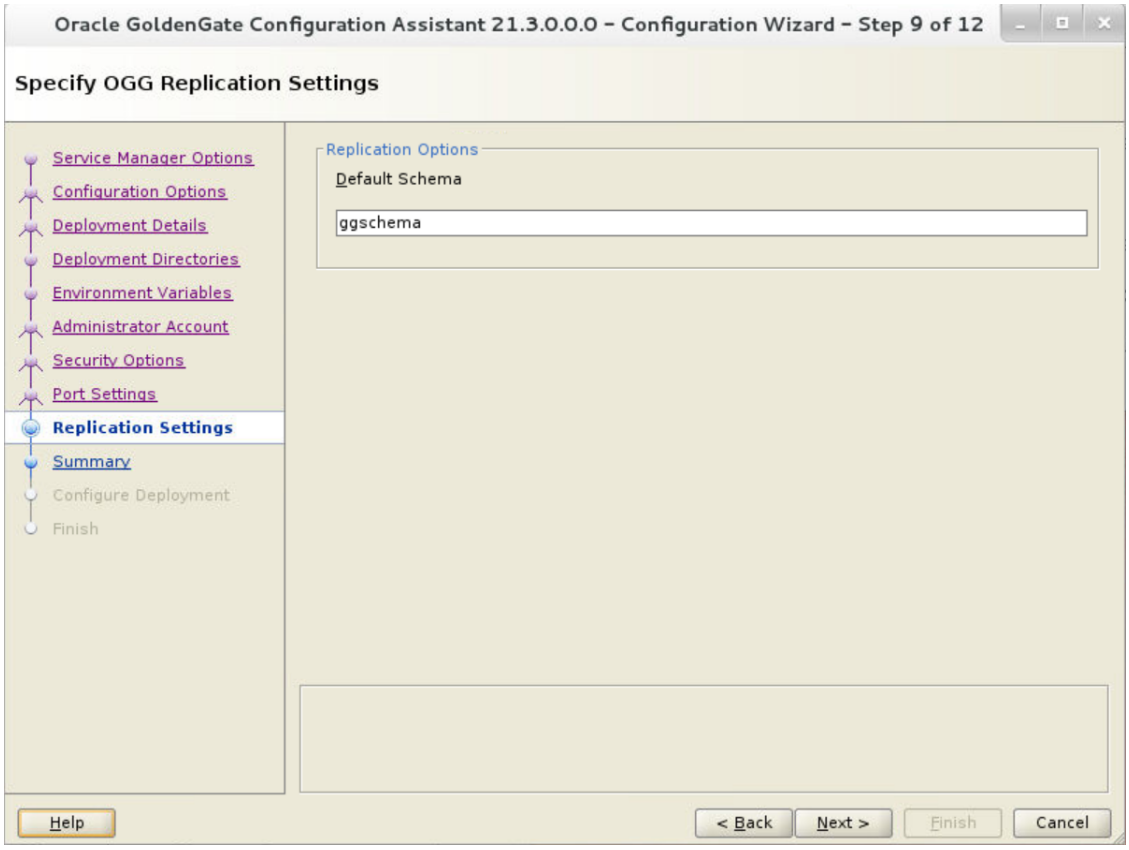
   Select the UDP port for performance monitoring. The option to select the UDP port is displayed only with deployments on Windows and other operating systems that don't support UDS communication with Performance Metric Service. See Protocols for Performance Monitoring for Different Operating Systems.

   You can change the TCP port from the Service Manager console after the deployment is done. For more information on `PMSRVR`, see ENABLEMONITORING.
4. Select the type of datastore as **Berkeley Database (BDB)**, which is the default or **Open LDAP Lightning Memory-Mapped Database** (LMDB).

   For learning more about BDB, see Oracle Berkeley DB 12c Release 1. For details on LMDB, see http://www.lmdb.tech/doc/.
5. You can also designate the Performance Monitor as a **Critical Service** if integrating the Service Manager with XAG.
6. Select the location of your datastore. BDB and LMDB are in-memory and disk-resident databases. The Performance Metrics Service uses the datastore to store all performance metrics information.
7. Click **Next**.

   The `oggca` utility validates whether or not the port you entered is currently in use or not.

Replication Settings

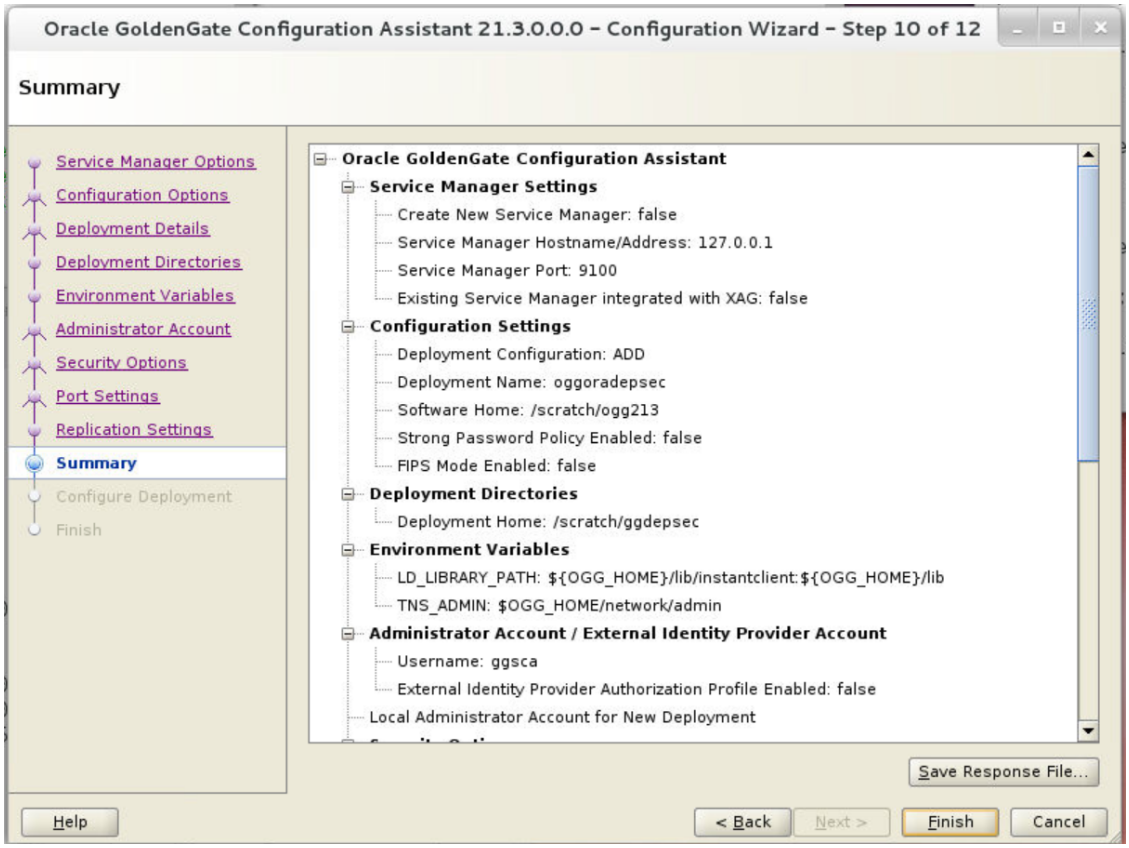


1. Enter the Oracle GoldenGate default schema that you want to use to store the replication objects such as checkpoint and heartbeat tables.

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

2. Click **Next**.

Summary



1. Review the detailed configuration settings of the deployment before you continue.
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** and then click **Next**.

Configure Deployment

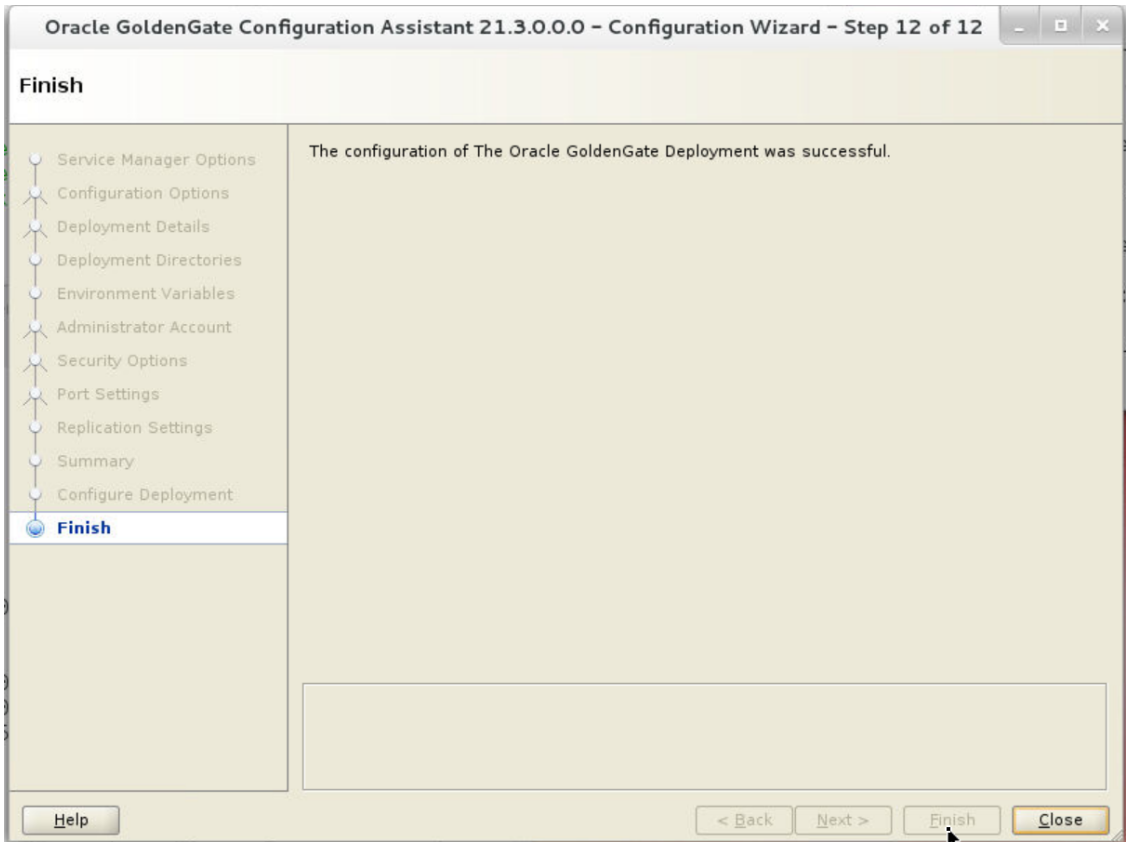
This screen displays the progress of the deployment creation and configuration. There could be some notifications during the progress if the Service Manager is registered as a service.

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

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

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

Finish



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

Using Silent Deployment

This section discusses the silent deployment process.

Add a Deployment in Silent Mode using OGGCA

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

- Open the Deployment response file template with extension .rsp, available at this location in Oracle GoldenGate:

  ogg-home/inventory/response

- Follow the instructions specified in the file to edit and then save the file with a different name, such as `oggca.rsp`.
- Create the Deployment by running the following command:

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

Example:

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

First Access to the Deployment from the Service Manager

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

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.

Add Users to a Deployment

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

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

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 **Administrator**.
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 3-1 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.

Table 3-1 (Cont.) Oracle GoldenGate User Roles and Privileges

Role ID	Privilege Level
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 Basic 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 **Administrator** 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.

Delegate User Authentication and Authorization to an External ID Provider

Learn about delegating user authentication and authorization to an external ID provider.

Topics:

Configure the Authorization Profile to Set Up IDCS Access Credentials

Oracle GoldenGate interoperates with external identity provider Oracle Identity Cloud Service (IDCS) for authentication and authorization of user credentials that are associated with your deployment.

After you set up the Oracle Identity Cloud Service (IDCS) user credentials in OGGCA on the Administrator Account screen, you need to perform these steps to set up an authorization profile for IDCS. This authorization profile will allow connecting and accessing the IDCS server to authorize users for Oracle GoldenGate.

To configure this type of user authentication and authorization, you need to create an authorization profile in Oracle GoldenGate.

Access the Authorization Profile

Use the following steps to set up this type of authorization profile for your deployment:

1. Click the **deployment** name or the **Service Manager** name from the Service Manager Overview page's **Deployment** section.
2. From the Deployment or Service Manager Details page, click the **Authorization Profiles** tab.
3. Click the plus sign (+) next to the **Profiles** section to start creating an authorization profile. Enter the following details for the profile:
   - 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)
4. 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. Here are the user role options that map the name of a group with respective role in IDCS:
   - Security Role
   - Administrator Role
   - Operator Role
   - User Role
5. Click **Submit** to create an authorization profile.
6. To enable the authorization profile for your deployment, select the authorization profile that you want to enable and click the **Enable Profile** toggle switch.

Manage Certificates for Deployments

Learn about managing certificates for deployments.

Topics:

Apply Certificates to an Oracle GoldenGate Deployment

Certificates can apply to:

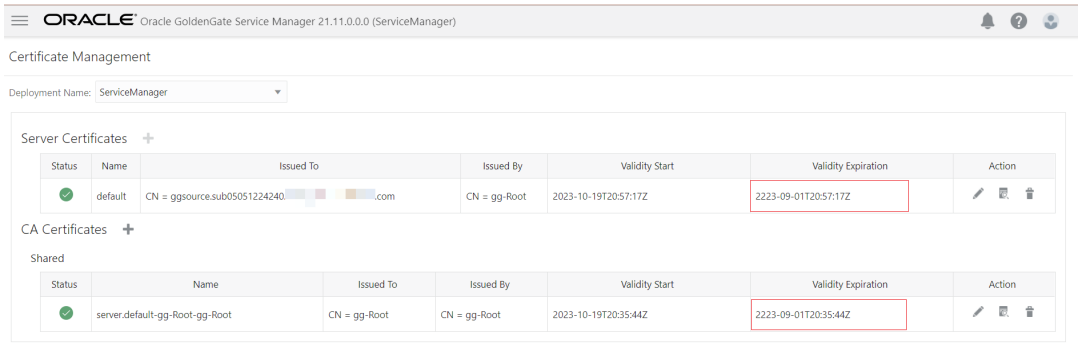
- **A specific deployment**: These certificates are local to the deployment. See Create Different Types of Certificates for a Secure Deployment.
- **Shared across deployments added to the same Service Manager**: These are shared certificates created from the Service Manager Certificate Management page. 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 a RootCA External Certificate in the Target Deployment. For more information, refer to Connecting Two Deployments Using External RootCA Certificate.

> **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.

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.

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**.

Modify Configuration for the Service Manager

Learn about how to modify configuration for the Service Manager.

Topics:

Access the Service Manager Information Page

To modify the Service Manager:

1. Select the **Service Manager** from the **Deployments** section of the Service Manager Overview page.
2. Use the Service Manager information page to edit the configuration for the Service Manager using these tabs. The tabs that you can edit for the Service Manager are the same as the tabs for the deployment.

   The following tabs can be used to set up options for the Service Manager:

Details Tab

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

For SQL Server and Db2 z/OS, you need to follow the steps given in the Setting up Environment Variables for Db2 z/OS, Setting up for DB2 z/OS, and Setting up for SQL Server sections in the *Using Oracle GoldenGate on Oracle Cloud Marketplace* guide.

> **Note:**
>
> It's important to do the settings for SQL Server and DB2 z/OS to make sure that the Administration Service starts when using either of these databases.

Configuration Tab

Use this tab to review and change the selected deployment environment variables. The environment variables that you configured with OGGCA are displayed. You can add new variables, modify existing variables, and delete selected variables. For Oracle database, make sure that `TNS_ADMIN` is set. See Specify Environment Variables for more information.

Certificates

Use this tab to add and manage certificates for the server, client and CA certificates. There is a difference between the Certificates tab and the Certificate Management page. The Certificates tab is associated with the deployment or Service Manager because you arrive at this tab by clicking the deployment name or the Service Manager. However, the Certificate Management page allows you to manage certificates by selecting the deployment or Service Manager on that page itself.

See Manage Certificates for Deployments for more information.

Authorization Profiles

Use this tab to delegate user and group management to external ID providers such as Oracle Identity Cloud Service (IDCS). Integration with an external Identity Management (IDM) system using OpenID/OAuth2.0 protocol provides Oracle GoldenGate users with:

- A single sign-on experience
- Ease of deploying Oracle GoldenGate cloud integration with IDCS.

See Delegate User Authentication and Authorization to an External ID Provider for more information.

Modify Configuration for the Deployment

Learn about how to modify the configuration for the deployment.

Topics:

Access the Deployment Information Page

To modify the Service Manager:

1. Select the **Service Manager** from the **Deployments** section of the Service Manager Overview page.
2. Use the Service Manager information page to edit the configuration for the Service Manager using these tabs. The tabs that you can edit for the Service Manager are the same as the tabs for the deployment.

The following tabs can be used to set up options for the Service Manager:

Details Tab

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

For SQL Server and Db2 z/OS, you need to follow the steps given in the Setting up Environment Variables for Db2 z/OS, Setting up for DB2 z/OS, and Setting up for SQL Server sections in the *Using Oracle GoldenGate on Oracle Cloud Marketplace* guide.

> **Note:**
>
> It's important to do the settings for SQL Server and DB2 z/OS to make sure that the Administration Service starts when using either of these databases.

Configuration Tab

Use this tab to review and change the selected deployment environment variables. The environment variables that you configured with OGGCA are displayed. You can add new variables, modify existing variables, and delete selected variables. For Oracle database, make sure that `TNS_ADMIN` is set. See Specify Environment Variables for more information.

Certificates

Use this tab to add and manage certificates for the server, client and CA certificates. There is a difference between the Certificates tab and the Certificate Management page. The Certificates tab is associated with the deployment or Service Manager because you arrive at this tab by clicking the deployment name or the Service Manager. However, the Certificate Management page allows you to manage certificates by selecting the deployment or Service Manager on that page itself.

See Manage Certificates for Deployments for more information.

Authorization Profiles

Use this tab to delegate user and group management to external ID providers such as Oracle Identity Cloud Service (IDCS). Integration with an external Identity Management (IDM) system using OpenID/OAuth2.0 protocol provides Oracle GoldenGate users with:

- A single sign-on experience
- Ease of deploying Oracle GoldenGate cloud integration with IDCS.

See Delegate User Authentication and Authorization to an External ID Provider for more information.

Manage the Status of Deployment and Microservices

Learn about managing the status of the deployment and the Microservices.

Topics:

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 Overview 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 of the Service Manager Overview page, locate the deployment that you need to start or stop.
3. From the **Action** column, you can select from the available options:
   - **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 the microservices associated with a deployment, 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 **Services** section of the Service Manager Overview page, go to the **Action** column for the specific microservice.
2. 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.

Manage the Microservices Configuration Details

Learn about managing the Microservices configuration details.

Topics:

View and Edit the Microservice Configuration

Use the Service Manager Overview page to **view** and **edit** the microservices configuration and restart options.

> **Note:**
>
> For all microservices, the configuration and restart options are the same but may have different values.

To access the configuration details of any of the microservices:

1. Click the **See Details** icon in the **Details** column of the Service section in the Service Manager Overview page. The Service Information page is displayed.
2. In the **Details** tab of the Service Information page, click the pencil icon in the **Service Configuration** section to edit the following configuration options for a microservice:
   - **Port**: Use this field to change the port number for the corresponding microservice.
   - **U-Mask**: File mode creation mask
   - **Config Force**: Use this toggle switch to enable or disable storing of configuration data forcefully.
   - **Enabled**: Displays if the microservice is enabled or not. You can choose the toggle switch to enable the microservice.
   - **Status**: Displays the status of the service.

View and Edit the Restart Options for Microservices

The restart options for a microservice allow you to define the behavior of a microservice, when it needs to restart. To modify the restart options from the Service Information page:

1. Click the pencil icon from the **Restart Options** section of the Service Information page.
2. View or edit the following restart options for the microservice:
   - **Enabled**: If set to true, then the service will attempt to restart automatically if it encounters an error.
   - **On Success**: If set to false, then the service is only restarted if it fails.
   - **Delay**: The time (in minutes) to pause between discovering that a process is terminated abruptly and restarting it.
   - **Retries**: The maximum number of trials to restart the service, before aborting the retry effort.

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

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.

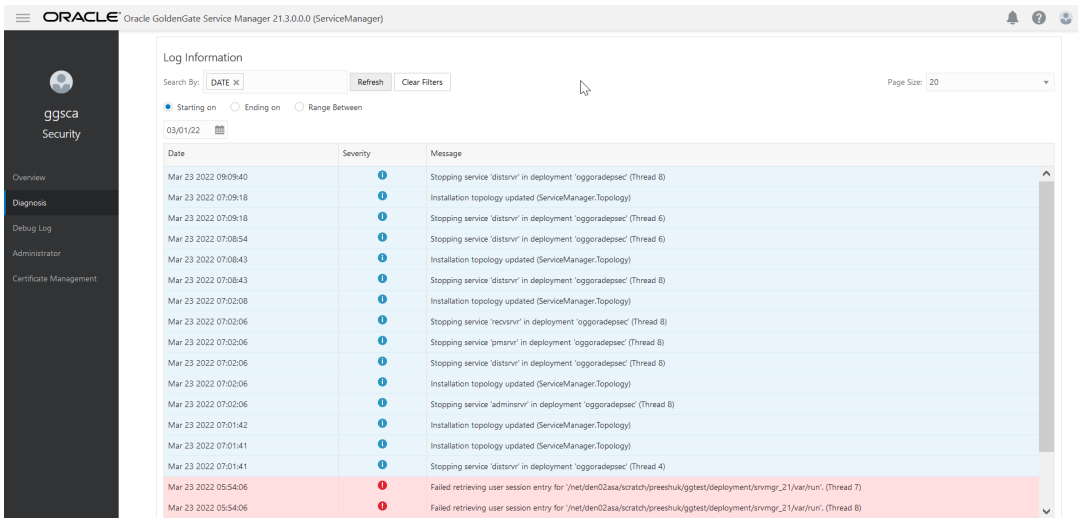
Access the **Diagnosis** page from the left navigation pane of the Service Manager. The complete log information is displayed on the page.

Search for Log Messages

If you want need to search for a specific message, you can also search for it by following these steps:

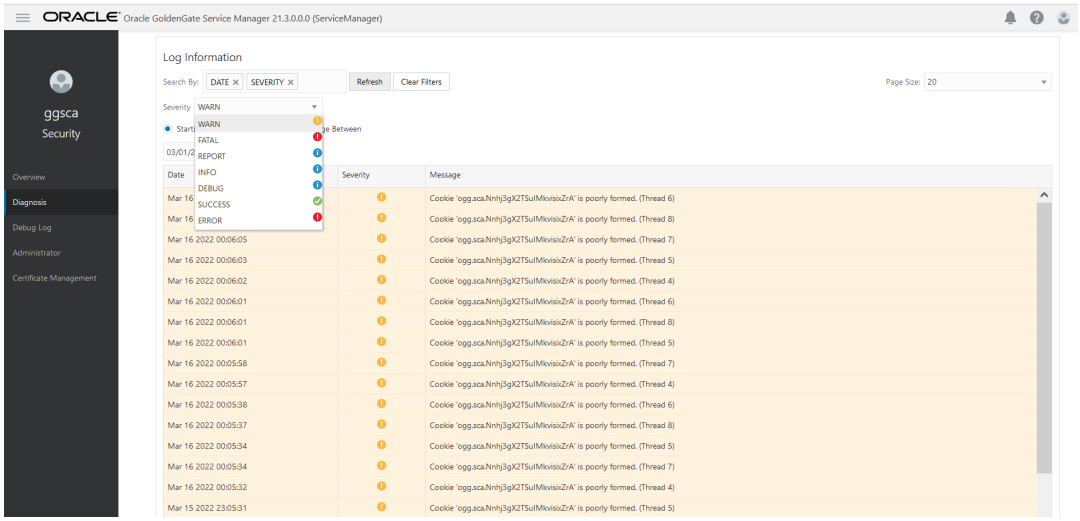
1. Enter a search criteria in the **Search** box. The search criteria can be **Date**, or **Severity** of the message(s), or the Message string itself. You can add multiple search criteria in the search box.
2. If you select **Date**, then you get the options:
   - Starting on: Displays the log information 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.



The screen shows the **Date** search criteria with the **Starting on** date.

3. If you select **Severity** of the message in the log, then you get to choose from the following levels of severity:



The screen shows various severity levels of messages. You can select any of these severity levels and search for log messages.

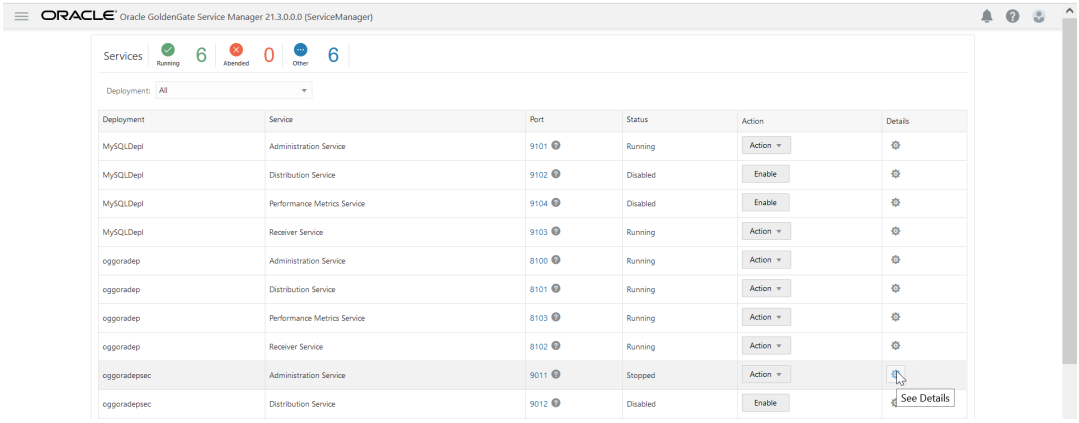
- Warn
- Fatal
- Report
- Info
- Debug
- Success
- Error

4. Click **Clear Filter** if you want to delete the search criteria.

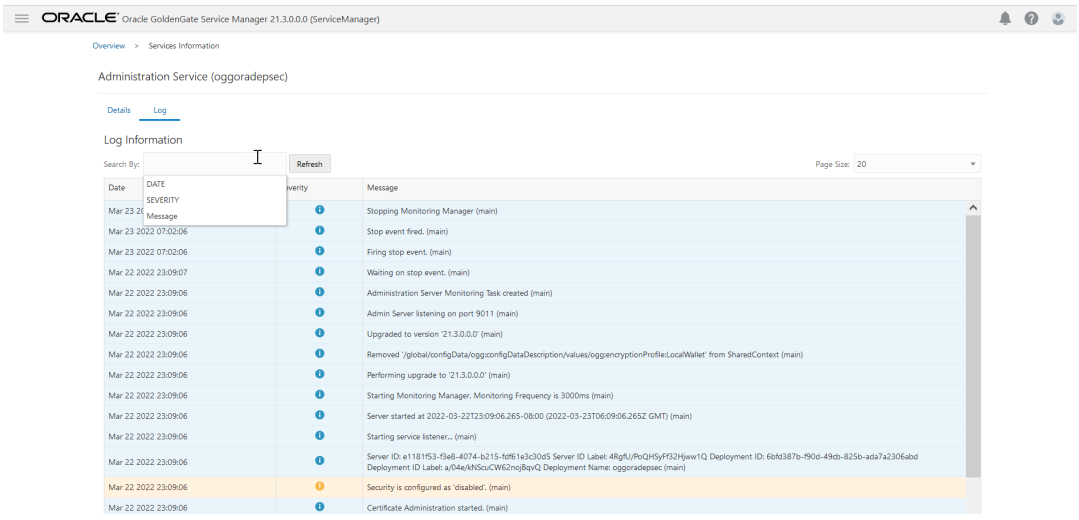
Search and Read Log Information for Microservices in a Deployment

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.

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.

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.
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 and click **Next**.
7. See the list of settings that are deleted with the deployment and click **Finish**.

Remove the Service Manager

Learn about removing the Service Manager.

Start OGGCA to Remove the Service Manager

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

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

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

2. Select **Existing Service Manager** from the **Select Service Manager Options** screen. Click **Next**.
3. Select the **Service Manager** from the drop down list.
4. Select **Remove Service Manager Deployment** from the **Configuration Options** screen.
5. Click **Finish** to remove the Service Manager.

Files to be Removed Manually After Removing Deployment

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

Operating System	Files to be Removed Manually to Unregister an Existing Service Manager
Linux 6 **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.	• `/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`

The following commands are executed to stop the Service Manager:

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

Note:

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

4
Prepare

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

Prepare Databases

Learn about preparing and configuring Oracle and non-Oracle databases for Oracle GoldenGate.

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 4-1 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

Table 4-1 (Cont.) Privileges Needed by Oracle GoldenGate for Db2 z/OS

User privilege	Extract	Stored Procedure	Replicat
SELECT ON the following SYSIBM tables: SYSTABLES SYSCOLUMNS SYSTABLEPART SYSKEYS SYSINDEXES SYSCOLAUTH SYSDATABASE SYSFOREIGNKEYS 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 DSNTIPE 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.

Specify the Path to the Initialization File

Specify the ODBC initialization file by setting the `DSNAOINI` environment variable in the z/OS UNIX profile, as in the following example:

```
export DSNAOINI="/etc/odbc810.ini"
```

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

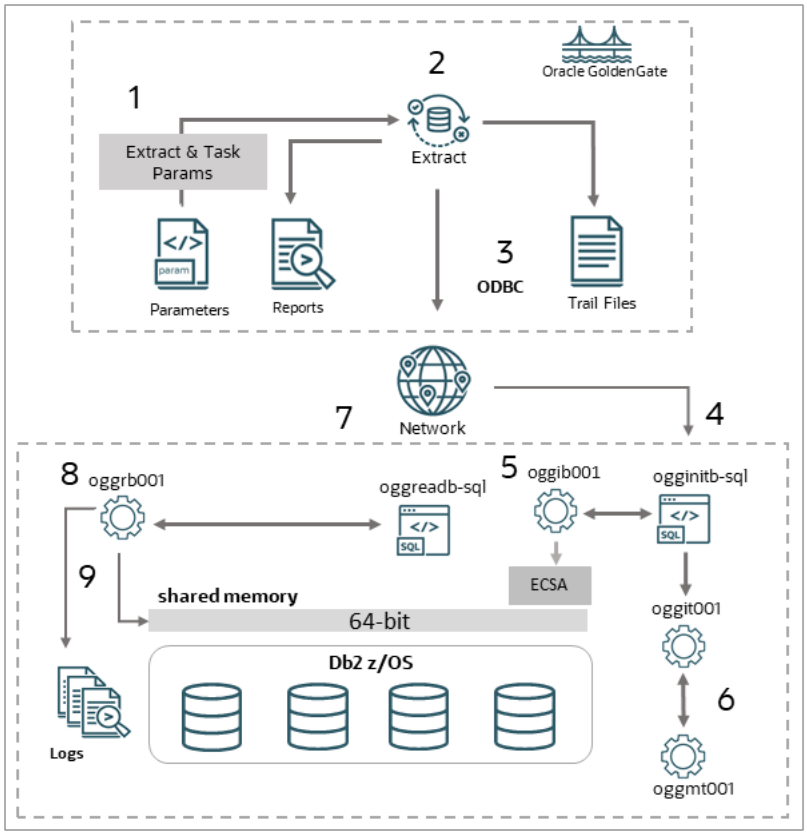
components consist of executable load modules, SQL stored procedures and functions, and external programs called via the stored procedures. These components are:

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 4-1 Replication Process for Db2 z/OS Extract



The process starts and runs as shown in the figure ablve, 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 RRSAF. Each time one of the Db2 WLM address spaces starts, it uses RRSAF 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 at `lib/utl/zOSutils.zip`, which contains a file called`zOSPrograms.zip`. Unzip `zOSPrograms.zip` to `zOSPrograms.tar` and copy `zOSPrograms.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 `zOSPrograms.tar`.

7. Restore the objects with the command: `tar -xovf zOSPrograms.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. The same script is restored through `zOSPrograms.tar` on the Db2 z/OS system in EBCDIC format and is suitable for use through native Db2 z/OS tools such as `SPUFI`.

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 4-2 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 `zOSPrograms.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.

Transaction Log Settings and Requirements

Know more about transaction log settings, requirements and the steps to add transaction logs for Oracle GoldenGate for Db2 z/OS.

Topics:

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 Db2 z/OS: Enable Change Capture

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 who has 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 also must have the `MONITOR2` privilege.

Size and Retain Logs

When tables are defined with `DATA CAPTURE CHANGES`, more data is logged 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 Extract for long periods of time.
- Your network is unreliable or slow.

To control log retention, use the `DSN6LOGP MAXARCH` system parameter in the `DSNTIJUZ` installation job.

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 thereafter.

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 over again.

> **Note:**
>
> The IBM documentation makes recommendations for improving 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, but it will degrade performance. For example, Db2 reserves taped archives for a single recovery task. Therefore, Extract would not be able to read an archive tape that is being used to recover a table until the recovery is finished. You could use DFHSM or an equivalent tools to move the archive logs in a seamless manner between online DASD storage and tape, but Extract will have to wait until the transfer is finished. Delays in Extract processing increase the latency between source and target data.

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 is used by Extract 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 its own 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. This includes `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. Unicode is not converted.
- 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. 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 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 objects that are 4K or smaller.
- Oracle GoldenGate supports the default `TIMESTAMP` and the `TIMESTAMP` with `TIMEZONE` to up to 9 digit fractional value, but no further.

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 in relation to 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.

- Extraction and replication of DML operations on Db2 for z/OS tables that contain rows of up to 512KB in length. This size exceeds the maximum row size of Db2.
- `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) and are specified 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 columns per table, which is supported by the database.

- Oracle GoldenGate supports the maximum column size that is supported by the database.
- Extraction and replication of data that is 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. To replicate a `TRUNCATE` , the Replicat process uses a `DELETE` operation without a `WHERE` clause.
- `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] |
  ```

  `transaction_ID` and `count` cannot be specified together.

  `transaction_ID` and `duration` cannot be specified together.
- Options supported with `SKIPTRANS` and `FORCETRANS`:

  ```
  SKIPTRANS transaction_ID
  [FORCE] FORCETRANS transaction_ID  [FORCE]
  ```

Non-Supported Objects and Operations for Db2 z/OS

The following objects and operations are not supported by Oracle GoldenGate on Db2 z/OS:

- Extraction or replication of DDL operations
- Clone tables
- Data manipulation, including compression, that is performed within user-supplied Db2 exit routines, such as:
  - Date and time routines
  - Edit routines (`CREATE TABLE EDITPROC`)
  - 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 Assign 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 support DDL replication, the MySQL user must have privileges to install the database plug-ins. The required permissions for the plug-in is only required with MySQL 5.7. The `INSERT` privilege is required on the `mysql.plugin` system table.
- 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. They must be specified in the Oracle GoldenGate parameter files with the `USERID` parameter.
- 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
`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

The MySQL command to grant these user privileges is:

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, PROCESS, REFERENCES,
INDEX, ALTER, SHOW
      DATABASES, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE,
REPLICATION SLAVE, REPLICATION
      CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE,
CREATE USER, EVENT, TRIGGER,
      CREATE ROLE, DROP ROLE ON *.* TO ggadmin WITH GRANT OPTION
```

Minimum User Privileges Required for Remote Capture

The minimum user privileges needed to run Oracle GoldenGate for MySQL remote capture are:

`SELECT`, `REPLICATION SLAVE`, `REPLICATION CLIENT`, and `SHOW VIEW`

The MySQL command to grant the minimum user privileges is:

```
GRANT SELECT, REPLICATION SLAVE, REPLICATION CLIENT, SHOW VIEW ON *.* TO
ggadmin
```

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 Credentials.

Configure the Database Connection

Oracle GoldenGate gets the name of the database it is supposed to connect to from the `SOURCEDB` parameter. To configure the connection for the `SOURCEDB` parameter, use the following format:

```
SOURCEDB dbname@hostname:port, USERID mysqluser, PASSWORD welcome
```

The `dbname` is the name of the MySQL instance, `hostname` is the name or IP address of the MySQL database server, `port` is the port number of the MySQL instance. If using an

unqualified host name, that name must be properly configured in the DNS database. Otherwise, use the fully qualified host name, for example `myhost.company.com`.

Configuring a Two-way SSL Connection in MySQL Capture and Delivery

To use the two way SSL in Oracle GoldenGate for MySQL capture and delivery, 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/5.7/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 Database Service, Amazon Aurora MySQL, Amazon RDS for MariaDB, Amazon RDS for MySQL, Azure Database for MySQL, Google Cloud SQL for MySQL, and MariaDB.

With Oracle GoldenGate release 21.10, SingleStoreDB and SingleStoreDB Cloud are now supported for delivery only, using the Oracle GoldenGate for MySQL Replicat.

From Oracle GoldenGate 21c (21.7) for MySQL 8.0 and higher, 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

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.
- All the components of Oracle GoldenGate for MySQL, including Extract, Replicat, and Admin Client connect to the database using the MySQL native API.
- 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 in non-transactional.

Database Character Set

MySQL provides a facility that allows users to specify different character sets at different levels.

Level	Example
Database	`create database test charset utf8;`
Table	`create table test( id int, name char(100)) charset utf8;`
Column	`create table test ( id int, name1 char(100) charset gbk, name2 char(100) charset utf8));`

Limitations of Support

- When you specify the character set of your database as utf8mb4/utf8, the default collation is `utf8mb4_unicode_ci`/`utf8_general_ci`. If you specify `collation_server=utf8mb4_bin`, the database interprets the data as binary. For example, specifying the `CHAR` column length as four means that the byte length returned is 16 (for `utf8mb4`) though when you try to insert data more than four bytes the target database warns that the data is too long. This is the limitation of database so Oracle GoldenGate does not support binary collation. To overcome this issue, specify `collation_server=utf8mb4_bin` when the character set is utf8mb4 and `collation_server=utf8_bin` for UTF-8.
- The following character sets are **not** supported:

  ```
  armscii8
  keybcs2
  utf16le
  geostd8
  ```

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`. Make certain that you specify a session character set in one of these ways when you configure Oracle GoldenGate.

Prepare Tables for Processing

This section describes how to prepare the tables for processing. Table preparation requires these tasks.

Topics:

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.

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

Know more about transaction log settings and requirements for Oracle GoldenGate for MySQL.

Topics:

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.

To determine where the Extract checkpoints are, use the `INFO EXTRACT` command. For more information, see INFO EXTRACT in *Command Line Interface Reference for Oracle GoldenGate*.

Setting Logging Parameters

To capture from the MySQL transaction logs, the Oracle GoldenGate Extract process must be able to find the index file, which contains the paths of all binary log files.

Extract expects that all of the table columns are in the binary log. As a result, `binlog_row_image` set as full is supported and this is the default. Other values of `binlog_row_image` are not supported.

> **Note:**
>
> Oracle recommends that the binary log is retained for at least 24 hours.

In MySQL 5.7, the `server_id` option must be specified along with `log-bin`, otherwise the server will not start. For MySQL 8.0, the `server_id` is enabled by default.

Extract checks the following parameter settings to get this index file path:

1. Extract `TRANLOGOPTIONS` parameter with the `ALTLOGDEST` option. If this parameter specifies a location for the log index file, Extract accepts this location over any default that is specified in the MySQL Server configuration file. When `ALTLOGDEST` is used, the binary log index file must also be stored in the specified directory. This parameter should be used if the MySQL configuration file does not specify the full index file path name, specifies an incorrect location, or if there are multiple installations of MySQL on the same machine. From Oracle GoldenGate 21c onwards, `ALTLOGDEST` parameter is optional for local Extract, however, for remote Extract this parameter is mandatory. When `ALTLOGDEST` is not specified, the binary log index and binary log filepaths will be fetched from the database directly. The paths thus fetched are also subject to same accessibilitychecks as in the existing process.

   To specify the index file path using `TRANLOGOPTIONS` with `ALTLOGDEST`, use a command similar to the following:

   ```
   TRANLOGOPTIONS ALTLOGDEST "/mnt/rdbms/mysql/data/logs/binlog.index"
   ```

   To capture from a remote server or in case of remote capture, you only need to specify the `REMOTE` option instead of the index file path on the remote server. For remote capture, specify the following in the Extract parameter file:

   ```
   TRANLOGOPTIONS ALTLOGDEST REMOTE
   ```

2. The MySQL Server configuration file: The configuration file stores default startup options for the MySQL server and clients. On Windows, the name of the configuration file is `my.ini`. On other platforms, it is `my.cnf`. In the absence of `TRANLOGOPTIONS` with `ALTLOGDEST`, Extract gets information about the location of the log files from the configuration file. However, even with `ALTLOGDEST`, these Extract parameters must be set correctly:

   - `binlog-ignore-db=oggddl`: This prevents DDL logging history table entries in the `binlog` and is set in the `my.cnf` or `my.ini` file.
   - `log-bin`: This parameter is used to enable binary logging. This parameter also specifies the location of the binary log index file and is a required parameter for Oracle GoldenGate, even if `ALTLOGDEST` is used. If `log-bin` is not specified, binary logging will be disabled and Extract returns an error.
   - `log-bin-index`: This parameter specifies the location of the binary log index. If it is not used, Extract assumes that the index file is in the same location as the log files. If this parameter is used and specifies a different directory from the one that contains the binary logs, the binary logs must not be moved once Extract is started.
   - `max_binlog_size`: This parameter specifies the size, in bytes, of the binary log file.

> **Note:**
>
> The server creates a new binary log file automatically when the size of the current log reaches the `max_binlog_size` value, unless it must finish recording a transaction before rolling over to a new file.

- `binlog_format`: This parameter sets the format of the logs. It must be set to the value of `ROW`, which directs the database to log DML statements in binary format. Extract silently ignores the `binlog` events that are not written in the `ROW` format instead of abending when it detects a `binlog_format` other than `ROW`.

> **Note:**
>
> MySQL binary logging does not allow logging to be enabled or disabled for specific tables. It applies globally to all tables in the database.

- `mysql.rds_set_configuration`: When capturing from MySQL Amazon RDS instance, you need to call the `mysql.rds_set_configuraton` stored procedure on MySQL command line, to retain the binary logs for a specific duration. By default, the default value of `binlog_retention_hours` for MySQL Amazon RDS is set to NULL, which implies that the binary logs are not retained.

  The following example shows the command to preserve the binary log for 24 hours:

  ```
  mysql > call mysql.rds_set_configuration('binlog retention hours', 24);
  ```

To locate the configuration file, Extract checks the *MYSQL_HOME* environment variable: If *MYSQL_HOME* is set, Extract uses the configuration file in the specified directory. If *MYSQL_HOME* is not set, Extract queries the `information_schema.global_variables` table to determine the MySQL installation directory. If a configuration file exists in that directory, Extract uses it.

3. For MariaDB version 10.2 and later, Oracle GoldenGate works in the same way as for MySQL but a new variable needs to be configured in the `my.cnf` or `my.ini` file. The variable that needs to be added is "`binlog-annotate-row-events=OFF`". Restart MariaDB after configuring this variable and then start the Extract process.

Changing the Log-Bin Location

Modifying the binary log location by using the *log-bin* variable in the MySQL configuration file might result in two different path entries inside the index file, which could result in errors. To avoid any potential errors, change the *log-bin* location by doing the following:

1. Stop any new DML operations.
2. Let the extract finish processing all of the existing binary logs. You can verify this by noting when the checkpoint position reaches the offset of the last log.
3. After Extract finishes processing the data, stop the Extract group and, if necessary, back up the binary logs.
4. Stop the MySQL database.
5. Modify the *log-bin* path for the new location.
6. Start the MySQL database.

7. To clean the old log name entries from index file, use `flush master` or `reset master` (based on your MySQL version).
8. Start Extract.

Capturing using a MySQL Replication Slave

You can configure a MySQL replication slave to capture the master's binary log events from the slave.

Typically, the transactions applied by the slave are logged into the relay logs and not into the slave's `binlog`. For the slave to write transactions in its `binlog`, that it receives from the master , you must start the replication slave with the `log-slave-updates` option as `1` in `my.cnf` in conjunction with the other binary logging parameters for Oracle GoldenGate. After the master's transactions are in the slave's `binlog` , you can set up a regular Oracle GoldenGate capture on the slave to capture and process the slave's `binlog`.

MySQL: Supported Data Types, Objects, and Operations

This section contains support information for Oracle GoldenGate on MySQL Database.

Oracle GoldenGate for MySQL supports capture and delivery of initial load and transactional data for supported MySQL database versions and supported variants, such as MariaDB, Amazon RDS for MySQL, Amazon Aurora MySQL, and Google Cloud SQL for MySQL.

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.

Topics:

Character Sets in MySQL

MySQL allows users to specify different character sets at different levels.

Level	Example
Database	`create database test charset utf8;`
Table	`create table test( id int, name char(100)) charset utf8;`
Column	`create table test ( id int, name1 char(100) charset gbk, name2 char(100) charset utf8));`

Limitations of Support

Oracle GoldenGate supports mixed character sets per listed objects, with the following limitations.

- 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
  utf8mb3
  ```

```
keybcs2
utf16le
geostd8
```

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
- 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 will abend.
- Table with single column is not supported for JSON datatype. Extract will abend 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 OGG-06556 The following columns will not be considered for CDR
  ```

  The limtations for CDR applies to cases where the `GETBEFORECOLS` and `COMPARECOLS` are used.

Non-Supported MySQL Data Types

Oracle GoldenGate for MySQL does not support the following data types:

All spatial types (Geometry and so on), `SET`.

> **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.
- Basic extraction and replication of DDL operations are supported for MySQL 5.7.10 and higher.
- For MySQL 5.7.10, only local DDL capture is supported.
- For MySQL 8.0, local and remote DDL capture is supported.
- Only the `CREATE TABLE`, `ALTER TABLE`, and `DROP TABLE` operations are 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, /* Geogian 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

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

Grant User Privileges for Oracle Database 21c and Lower

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

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

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

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

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

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

Privilege	Extract	Replicat All Modes	Purpose
`RESOURCE`	Yes	Yes	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;`

Privilege	Extract	Replicat All Modes	Purpose
`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 mutitenant 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 *`GGADMIN`* should be a valid Oracle GoldenGate administrator schema.
`EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE ('REPUSER', CONTAINER=>'PDBEAST');`	Yes	Yes	• Required for Autonomous Databases (ATP and ADW) Extract and Replicat. Extracts in the root container (`CDB$ROOT`)) might require a value of `ALL` or a specific PDB (example: `pdbeast`). • Grant privileges for Extract and Replicat users. See Example: Grant privileges using the DBMS_GOLDENGATE_AUTH .GRANT_ADMIN_PRIVILEGE package • Grant privileges to capture from Virtual Private Database • Grants privileges to capture redacted data
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

Privilege	Extract	Replicat All Modes	Purpose

`GRANT INSERT ANY TO...`
 	
`GRANT UPDATE ANY TO...`
 	
`GRANT DELETE ANY TO...`
	NA	Yes	Grant these privileges to the Replicat user, instead of granting `INSERT`, `UPDATE`, `DELETE` to every table, if replicating every table.

If DDL replication is performed, grant the following as Database Vault owner: `EXECUTE DBMS_MACADM.AUTHORIZE_DDL('GGADMIN USER', 'SCHEMA FOR DDL');`	No	No	Capture from Data Vault. See Privileges for Capturing from Oracle Data Vault.
DDL privileges on target objects (if using DDL support)	NA	Yes	Issue replicated DDL on target objects. See Details of Support for Objects and Operations in Oracle DDL.
`GRANT [CREATE|ALTER|DROP] ANY [TABLE|INDEX|VIEW| PROCEDURE] to GGADMIN;`	Yes	Yes	Grants privileges for DDL Replication for tables.
`CREATE ANY TABLE`	Yes	Yes	Grants privileges for creating table in any schema. To allow creating tables only in a specific schema, use the `CREATE TABLE` privilege.
`CREATE ANY VIEW`	Yes	Yes	Grants privileges to create view in any database schema. To allow creating views in a specific schema, use the `CREATE VIEW` privilege.
`SELECT ANY DICTIONARY`	Yes	Yes	Allow all privileges to work properly on dictionary tables.

Example: Permissions granted for the Oracle database common user

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

```
CREATE USER c##ggadmin IDENTIFIED BY passw0rd CONTAINER=all DEFAULT
TABLESPACE GG_DATA TEMPORARY TABLESPACE temp;
GRANT RESOURCE to c##ggadmin;
GRANT CREATE SESSION to c##ggadmin;
GRANT CREATE VIEW to c##ggadmin;
GRANT CREATE TABLE to c##ggadmin;
GRANT CONNECT to c##ggadmin CONTAINER=all;
```

```
GRANT DV_GOLDENGATE_ADMIN; --- for data vault user
GRANT DV_GOLDENGATE_REDO_ACCESS; --- for data vault user
GRANT ALTER SYSTEM to c##ggadmin;
GRANT ALTER USER to c##ggadmin;
ALTER USER c##ggadmin SET CONTAINER_DATA=all CONTAINER=current;
ALTER USER c##ggadmin QUOTA unlimited ON GG_DATA;
GRANT SELECT ANY DICTIONARY to c##ggadmin;
GRANT SELECT ANY TRANSACTION to c##ggadmin;
EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE('c##ggadmin');
```

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

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

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

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

Example: Grant privileges using the DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE package

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

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

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

Privileges for Capturing from Oracle Data Vault

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

- ```
  EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE ('userID','*',
  GRANT_OPTIONAL_PRIVILEGES=>'*');
  GRANT DV_GOLDENGATE_ADMIN, DV_GOLDENGATE_REDO_ACCESS to userID;
  ```

- Grant Replicat the privileges in `DBMS_MACADM.ADD_AUTH_TO_REALM` if applying to a realm.

  Connect as Database Vault owner and execute the following scripts:

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

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

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

Prepare Database Connection, System, and Parameter Settings

Learn about configuring database connection, system, and parameter settings for Oracle GoldenGate for Oracle.

Enable Oracle GoldenGate for Oracle

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

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

```
ENABLE_GOLDENGATE_REPLICATION=true
```

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

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
STATS EXTRACT command with REPORTFETCH option	Shows Extract fetch statistics on demand.
STATOPTIONS parameter with REPORTFETCH option	Sets the STATS EXTRACT command so that it always shows fetch statistics.
MAXFETCHSTATEMENTS parameter	Controls the number of open cursors for prepared queries that Extract maintains in the source database, and also for SQLEXEC operations.
MAXFETCHSTATEMENTS parameter	Controls the default fetch behavior of Extract: whether Extract performs a flashback query or fetches the current image from the table.
FETCHOPTIONS parameter with the USELATESTVERSION or NOUSELATESTVERSION 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.
REPFETCHEDCOLOPTIONS 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.

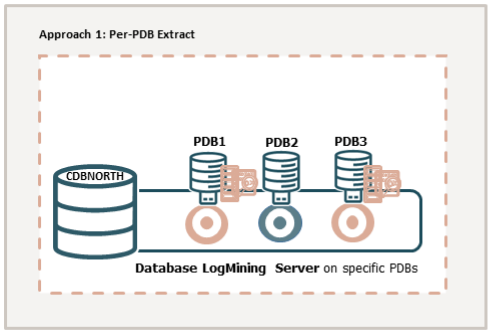
The following table lists the database properties and the associated concern/resolution.

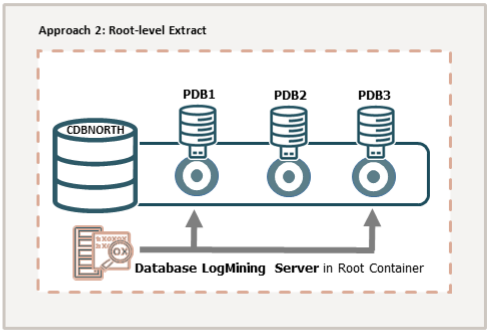
Database Property	Concern/Resolution
Table with interval partitioning	To support tables with interval partitioning, make certain that the `WILDCARDRESOLVE` parameter remains at its default of `DYNAMIC`.
Table with virtual columns	Virtual columns are not logged, and Oracle does not permit DML on virtual 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 `TABLE` statement and use the `FETCHCOLS` option of `TABLE` to fetch the value from the virtual column in the database. In the Replicat `MAP` statement, map the source virtual column to the non-virtual target column.
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. For more information, see *Parameters and Functions Reference for Oracle GoldenGate*.
`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 *not* require the Oracle GoldenGate DDL support environment. For more information, see *Parameters and Functions Reference for Oracle GoldenGate*.

Configure a Multitenant Container Database

Oracle GoldenGate with Oracle Database allows each pluggable database (PDB) to have Extract registered for a specific PDB, which is called a per-PDB Extract.

The following diagram shows the configuration for different approaches in a multitenant container database configuration:





There are two approaches to configure an Extract for an Oracle multitenant database.

- **Approach 1:** Adding Extract directly from the PDB. This approach is useful when Extract captures from isolated PDBs, managing ownership and responsibility at the PDB level.
- **Approach 2:** Adding Extract in the root container and referencing the associated PDBs This approach is useful when Extract captures data from multiple PDBs.

If you use **Approach 1**, you can create a per-PDB Extract by connecting as the local PDB user (for example, `ggadmin`) and then register this Extract with the PDB. As you are already logged in as the PDB user, the `container` clause is *not* required. Similarly, the `SOURCECATALOG` and two-part naming convention is adequate.

If you use **Approach 2**, you can connect to the root container with the common database user `c##ggadmin` and create Extract for specific PDBs. This Extract needs to be registered for the specific PDB using the container clause. The container clause might contain a single or multiple PDBs.

> **Note:**
>
> Even if you use a root-level Extract, you need the user credentials for each PDB from which you need to capture. The hearbeat table also resides in the individual PDBs.

To set up an Extract, see Add a Primary 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.
- To add a user for the root container, Extract must connect to the root container (`cdb$root`) as a common user in order to interact with the logmining server. To specify the root container, use the appropriate SQL*Net connect string for the database user that you specify with the `USERIDALIAS` parameter, such as `c##ggadmin@dbeast`.

- 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 Credentials to add a multitenant container database user in Oracle GoldenGate credentials. See Grant User Privileges for Oracle Database 21c and Lower 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	`updateSequence`	Grants `EXECUTE` to the owner of the Oracle GoldenGate DDL objects, or other selected user if not using DDL support.
Target	`replicateSequence`	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. The `GLOBALS` file must contain a `GGSCHEMA` parameter that specifies the schema in which the procedures are installed. This user must have `CONNECT`, `RESOURCE`, and `DBA` privileges.
3. Before using `FLUSH SEQUENCE`, issue the `DBLOGIN` command as the database user that has `EXECUTE` privilege on the `updateSequence` procedure. If logging into a multitenant container database, log into the pluggable database that contains the sequence that is to be flushed.

`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 for sequences: ggate	PDB User: ggadmin

```
sqlplus / as sysdba
ALTER SESSION 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 / as sysdba
ALTER SESSION SET CONTAINER=DBEAST;
@sequence.sql
```

When prompted enter the following:

```
GGADMIN GLOBALS
GGSCHEMA GGADMIN
```

Run the `FLUSH SEQUENCE` command:

```
DBLOGIN USERIDALIAS ggeast DOMAIN OracleGoldenGate
FLUSH SEQUENCE DBEAST.HR.*
```

Target Oracle GoldenGate Configuration:

```
sqlplus / as sysdba
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 upgrateding 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 the these servers.

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

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

```
3 GB * 1.25 = 3.75 GB
```

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 Database 23c and higher or through the procedure call `DBMS_GOLDEN_GATE_AUTH.GRANT_ADMIN_PRIVILEGE` for earlier database versions.

The following example shows how to login to a CDB as the system user and run the `sequence.sql` script:

```
sqlplus system/***@cdb23_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 Configure Logging Properties).

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

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

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

> **Note:**
>
> `ALLOWNULLABLEKEYS` is not valid for integrated Replicat.

4. 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*.

Configure Logging Properties

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

This section addresses the following logging levels that apply to Oracle GoldenGate. 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	`ALTER DATABASE ADD SUPPLEMENTAL LOG DATA`	Enables minimal supplemental logging to add row-chaining information to the redo log.	Required for all Oracle GoldenGate use cases
Schema-level supplemental logging, default setting See Enable Subset Database Replication 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 *scheduling columns*.	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.	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.
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.	Use only for nonintegrated 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 *scheduling columns*.	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`.

Logging option	Command Name	What it does	Use case
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 nonintegrated Replicat and non-parallel Replicat. This is the minimum required table-level logging.

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.

To know more, see `ALTER DATABASE` in the *Oracle Database SQL Language Reference*.

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 database when using the Oracle GoldenGate DDL replication feature. In all other use cases, it is optional, but then you must use table-level logging instead (see 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.

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.

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

Find out the supported data types, objects, and operations for Oracle GoldenGate on Oracle Database.

Details of Support for Oracle Database Editions

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

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

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

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

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

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

Details of Support for Oracle Data Types and Objects

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.

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

Detailed support information for Oracle data types, objects, and operations starts with 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`
- `CHAR`, `VARCHAR2`, `LONG`, `NCHAR`, `NVARCHAR2`
- `RAW`, `LONG RAW`, `CLOB`, `NCLOB`, `BLOB`, `SECUREFILE`, `BASICFILE`, and `BFILE` (LOB size limited to 4GB)

- XML columns stored as CLOB, Binary and Object-Relational (OR)
- XMLType columns and XMLType tables stored as XML CLOB, XML Object Relational, and XML Binary
- Native JSON datatype identified by the DTYJSON code.
- 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 it's 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 paramater must not be set for VARCHAR2, NVARVAR2, 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
(N)CHAR, (N) VARCHAR2 LONG, RAW, LONG RAW (N)CLOB, CLOB, BLOB, SECUREFILE, BASICFILE and BFILE	Not supported
XML columns, XMLType	Not supported
Native JSON datatype	VARCHAR2, NVARCHAR2, CLOB, NCLOB not supported with the Extract parameter binary_json_format.
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.

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.

Topics:

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.

Topics:

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 Setting Flashback Query.

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 `OCIString` object of the Oracle Call Interface, which does not support `NCHAR`, `NVARCHAR2`, or `NCLOB` attributes, so Replicat must bind them as `CHAR`. Connecting to the target in `AL32UTF8` prevents data loss in this situation. `HAVEUDTWITHNCHAR` must 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 index

  Table where the combination of all scalar columns do not guarantee uniqueness are unsupported.
- Tables with the following XML characteristics:
  - Tables with a primary key constraint made up of XML attributes
  - XMLType tables with a primary key based on an object identifier (PKOID).
  - XMLType tables, where the row object identifiers (OID) do not match between source and target
  - XMLType tables created by an empty CTAS statement.
  - XML schema-based XMLType tables and columns where changes are made to the XML schema (XML schemas must be registered on source and target databases with the `dbms_xml` package).
  - The maximum length for the entire `SET` value of an update to an XMLType larger than 32K, including the new content plus other operators and XQuery bind values.
  - SQL*Loader direct-path insert for XML-Binary and XML-OR.
- Tables with following UDT characteristics:
  - UDTs that contain CFILE or OPAQUE (except of XMLType)
  - UDTs with CHAR and VARCHAR attributes that contain binary or unprintable characters
  - UDTs using the RMTTASK parameter
- UDTs and nested tables with following condition:
  - Nested table UDTs with CHAR, NVARCHAR2 or NCLOB attributes.

  - Nested tables with CLOB, BLOB, extended (32k) VARCHAR2 or RAW attributes in UDTs.
  - Nested table columns/attributes that are part of any other UDT.
- When data in a nested table is updated, the row that contains the nested table must be updated at the same time. Otherwise there is no support.
- When VARRAYS and nested tables are fetched, the entire contents of the column are fetched each time, not just the changes. Otherwise there is no support.
- Object table contains the following attributes:
  - Nested table
  - SDO_TOPO_GEOMETRY
  - SDO_GEORASTER

See additional exclusions in Details of Support for Oracle Data Types and Objects.

Details of Support for Objects and Operations in Oracle DML

Here is a list of Oracle objects and operations that Oracle GoldenGate supports for the capture and replication of DML operations.

Topics:

Multitenant Container Database

Oracle GoldenGate captures from, and delivers to, a **multitenant container database**. See Configure a Multitenant Container Database.

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

Tables, Views, and Materialized Views

Oracle GoldenGate supports the following DML operations made to 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. 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`.

Topics:

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, the source database compatibility must be set to 11.2.0.0.0 or higher.

- Oracle GoldenGate supports Transparent Data Encryption (TDE).
- 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
  - `SQLLDR` with `DIRECT=TRUE`
- Oracle GoldenGate fully supports capture from compressed objects for Extract.
- Oracle GoldenGate supports XA and PDML distributed transactions.
- 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.

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. From Oracle GoldenGate 21c onward, a Partition Name Record (PNR) is generated for system partitioned tables, if it is included in the `PARTITION` parameter.

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).

Topics:

Limitations of Support for Sequences

These limitations apply to Extract.

- Oracle GoldenGate does not support the replication of sequence values in an active-active bi-directional configuration.
- The cache size and the increment interval of the source and target sequences must be identical. The cache can be any size, including 0 (`NOCACHE`).
- The sequence can be set to cycle or not cycle, but the source and target databases must be set the same way.
- Tables with default sequence columns are excluded from replication for Extract.

Non-supported Objects and Operations in Oracle DML

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

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

DML Auto Capture

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

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

Details of Support for Objects and Operations in Oracle DDL

Learn about the Oracle objects and operations that Oracle GoldenGate supports for the capture and replication of DDL operations.

Supported Objects and Operations in Oracle DDL

DDL capture support is integrated into the database logmining server. You must set the database parameter compatibility to 11.2.0.4.0 or higher. 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;`

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 Extract for the following Oracle Database objects:
  - functions
  - library
  - packages (specification and body)
  - procedure
  - synonyms
  - types (specification and body)
  - views
- From Oracle GoldenGate 21c onward, DDLs that are greater than 4 MB in size will be provided replication support.
- 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 dictionary for use with `NOUSERID` and `TRANLOGOPTIONS GETCTASDML`. This means that Extract receives object metadata from the LogMiner dictionary without querying the dictionary objects. Oracle GoldenGate uses the dictionary

automatically when the source database compatibility parameter is greater than or equal to 11.2.0.4.

When using dictionary and trail format in the Oracle GoldenGate release 12.2.*x*, Extract 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 Extract. 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.

- Starting with Oracle database 21c, the following DDL is available to support blocking of DML/DDL changes that are not replicated by Oracle GoldenGate:

  ```
  ALTER DATABASE [ENABLE | DISABLE] goldengate blocking mode;
  ```

  When Oracle GoldenGate blocking mode is enabled, DMLs that use `support_mode NONE` in tables and execute unsupported Oracle PL/SQL statements will fail with the following error:

  ```
  ORA-26981: "operation was unsupported during Oracle GoldenGate blocking
  mode"
  ```

  For Oracle database 21c, the following features cause a table to have `support_mode NONE` in Oracle GoldenGate:

  - `BFILE` as an attribute of ADT column, or typed table
  - Table with no scalars
  - OLAP AW$ table
  - Sharded queue table
  - Sorted Hash Cluster Table
  - Primary key constraint on ADT attribute in relational table
  - Primary key/unique key constraint on long `raw`/`varchar` (over 4000 bytes)

  - `V$DATABASE` column, `Goldengate_Blocking_Mode` can be queried to determine the current blocking mode status.

- For DDL auto capture mode:
  - It is relevant only for `DDL INCLUDE MAPPED` because Extract captures DDLs based on `TABLE` and `TABLEEXCLUDE` parameter.
  - Only table-related DDLs can be auto-captured.
  - DDLs to enable auto capture at table level:

    ```
    CREATE/ALTER TABLE … ENABLE LOGICAL REPLICATION ALLKEYS;
    ```

    or

    ```
    CREATE/ALTER TABLE … ENABLE LOGICAL REPLICATION ALLOW NOVALIDATE KEYS;
    ```

    See How to Capture Supplemental Logging for Oracle GoldenGate in *Oracle Database Utilities* guide.

- The following operations are supported for partition related DDLs and partition maintenance operations
  - Drop partition:

    If a partition is recreated with the same name, then it will get a new object number. The internal caches are cleared to minimize space consumption when a drop partition DDL is processed.
  - Truncate partition:

    Partition name and object number stays the same. Base table object version stays the same.
  - Rename partition:

    The partition object number stays the same but gets a new name. The base table's object version gets bumped. In memory name cache will get invalidated upon seeing this DDL and repopulated upon the next DML. The cache, which stores if a given partition object number is interesting or not will also need to be reevaluated as a the new partition name may switch from filtered to not filtered or vice versa.
  - Exchange partition:

    Exchanges data in a partition with that in a table or vice versa. The obj# of the partition being exchanged does not change. Dataobj# does change but is not used by Extract. The partition itself still belongs to the same table.
  - Merge partition:

    Merges one or more partitions into a new partition. The DDL creates the new partition and drops the partitions from which it was merged. In memory caches should be cleared to save space and the user should ensure proper filter rules for the newly created partition.
  - Split partition:

    The partition being split keeps its original name and object number and new partition is created for the split data. The user must ensure partition filter rules are correct for the newly created partition.
  - Coalesce partition:

Reduces the number of partitions in a hash partitioned table. The specific partition that is coalesced is selected by the database, and is dropped after its contents have been redistributed. The remaining partitions keep their same name and object number. The internal caches should be cleared to minimize space consumption.

- Modify partition:

  Modifies default and real attributes of partitions, apart from adding or dropping of values for list partitions. All modifications leave the partitions name and object number intact.

- Move partition:

  Partition data is moved to a new tablespace. Partition name and number remain the same.

- Redef table:

  `dbms_redefinition` can be used to partition a table through the use of an interim table. The partitions are created on the interim table and after the `finish_redef` operation, the tables swap names. The partitions created on the interim table keep their names and object numbers when the tables are swapped. The Extract filter cache, needs to be reevaluated upon `finish_redef` as the partitions now belong to the base table. The user must ensure proper filter rules.

- Redef partition:

  When redefining a table, the partitions follow from the original table to the interim table. For example, consider the case where the original table has partitions, which live in the `USER` tablespace, and the interim table is created with no partitions and the table lives in the `NEW` tablespace. In this case, after the `finish_redef` operation, when the tables are swapped the partition still lives in the `USER` tablespace. Redef partition allows a partition to be moved to the interim table's `NEW` tablespace. The partition retains its name and object number.

- System generated partition names:

  When partitions are created automatically for hash partitions and operations such as split partition, the partition name is in the form of `SYS_P` *`sequence value`*. Similarly, subpartitions are of the form `SYS_SUBP` *`sequence value`*. It is recommended that the partition is renamed before excepting DML to conform to filter rules.

Non-supported Objects and Operations in Oracle DDL

Here's a list of non-supported objects and operations in Oracle DDL.

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", // Adminstrator 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*",
"*.SYS_EXPORT_TABLE*",
```

```
"*.CMP*$*", // space management, rdbms >= 12.1
"*.DBMS_TABCOMP_TEMP_*", // space management, rdbms < 12.1
"*.MDXT_*$*" // Spatial extended statistics tables
```

Other Non-supported DDL

Oracle GoldenGate does not support the following:

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

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

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

Topics:

Database 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			

Privilege	Extract	Replicat	Purpose
`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;`
`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 `GGSCHEMA` 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;`

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 Credentials topic to know how to create database connections.

Configuring a Database Connection

Oracle GoldenGate connects to a PostgreSQL database through an ODBC (Open Database Connectivity) driver and requires a system Data Source Name (DSN) be created with the correct database connection details for each source and target PostgreSQL database.

This section contains instructions for setting up the DSN connections that Extract and Replicat will use.

Ensure that you have installed and configured the driver prior to creating a DSN, by following the Installing the DataDirect driver for PostgreSQL instructions.

> **Note:**
>
> Do not use `PgBouncer` setup for Extract connections to the PostgreSQL database because `PgBouncer` does not understand the replication protocol, because of which the Extract connection is not identified as replication connection.

Configuring a Database Connection in Linux

To create a database connection in Linux for Oracle GoldenGate processes, create a DSN (Data Source Name) inside the `/etc/odbc.ini` file. Multiple DSNs can be part of the same ODBC 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 value of the Oracle GoldenGate installation path, for example: `/u01/app/ogg`.
- `Driver`: For Oracle GoldenGate release versions prior to 21.8, set to `/GoldenGate_Installation_Path/lib/GGpsql25.so`.

  For Oracle GoldenGate release versions 21.8 and later, set the value to `/<GoldenGate_Installation_Path>/datadirect/lib/ggpsql25.so`.
- `Database`: Is the name of the source or target database.
- `HostName`: Is the database host IP address or host name.
- `PortNumber`: Is the listening port of the database.
- 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 a sample `/etc/odbc.ini` file with two DSN entries. The Data Source names used in the example below are `PG_src` and PG_tgt.

1. Create a DSN for each source or target database in the `/etc/odbc.ini` file.

```
sudo vi /etc/odbc.ini
```

```
#Sample DSN entries [ODBC Data Sources]
PG_src=Oracle GoldenGate PostgreSQL Wire Protocol
PG_tgt=Oracle GoldenGate PostgreSQL Wire Protocol
[ODBC] IANAAppCodePage=4 InstallDir=/u01/app/ogg
```

```
[PG_src]
Driver=/u01/app/ogg/datadirect/lib/ggpsql25.so
Description=Oracle GoldenGate PostgreSQL Wire Protocol
Database=sourcedb
HostName=remotehost
PortNumber=5432
```

```
[PG_tgt]
Driver=/u01/app/ogg/datadirect/lib/ggpsql25.so
Description=Oracle GoldenGate PostgreSQL Wire Protocol
Database=targetdb
HostName=remotehost
```

```
PortNumber=5432
```

2. Save and close the `odbc.ini` file.

> **Note:**
>
> You can enable `ODBC` tracing without changing the `ODBCINI` file using the ODBCTRACE parameter. See `ODBCTRACE` in *Parameters and Functions Reference for Oracle GoldenGate*.

Configuring a Database Connection in Windows

To create a database connection in Windows, use the Windows ODBC Data Source Administrator to create a system DSN for each source and target database.

1. On the Windows system, open the Control Panel folder.
2. Open the Administrative Tools folder.
3. Open ODBC Data Sources (64-bit). The ODBC Data Source Administrator dialog box is displayed.
4. Select the System DSN tab, and then click **Add**.
5. Under Create New Data Source, select the Oracle GoldenGate PostgreSQL Wire Protocol driver and click **Finish**.
6. The Create a New Data Source wizard is displayed.
7. Supply the following:
   - For Data Source Name, type a name for the DSN, up to 32 alpha-numeric characters in length, excluding special keyboard characters except for the underscore and dash.
   - (Optional) For Description, type a description of this DSN.
   - Provide the database server's Host Name, the database Port Number, and Database Name.
8. Click **OK** to close the dialog box.

   You can also provide the User Name information under the Security tab but it is recommended instead to leave this field empty and instead store the user name and password in the Oracle GoldenGate wallet as a credential alias, and reference them with the `USERIDALIAS` parameter in Extract and Replicat.

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 $ODBCINI 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 below mentioned ODBC DSN attributes should be setup accordingly in `$ODBCINI`.

```
KeyStore=<path to .p12 keystore file> KeyStorePassword=<keystore-passwd>
TrustStore=<path to root certificate> ValidateServerCertificate=1
```

> **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`:

- 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, GoogleAlloyDB 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 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.

Prepare Tables for Processing

You must perform the following tasks to prepare your tables for use in an Oracle GoldenGate environment for PostgreSQL.

Topics:

Disabling Triggers and Cascade Constraints on the Target

If Oracle GoldenGate is configured to capture DML operations from source tables that occur due to trigger operations or cascade constraints, then disable the triggers and cascade delete and cascade update constraints on the target tables.

If not disabled, the same trigger or constraint gets activated on the target table and becomes redundant because of the replicated data. 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.

In the Replicat `MAP` statements, map the source tables to appropriate targets, and map the child tables that the source tables reference with triggers or foreign-key cascade constraints. Triggered and cascaded child operations must be mapped to appropriate targets to preserve data integrity. Include the same parent and child source tables in the Extract `TABLE` parameters.

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. For PostgreSQL LOB types such as `text`, `xml`, `bytea`, `char`, `varchar`, Oracle GoldenGate supports these columns as a primary key in source or target tables up to a length of 8191 bytes.

   > **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.
   > For tables that have no uniqueness and have repeat rows with the same values, Replicat will Abend on update and delete operations for these rows.

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*.

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.

- Capture and delivery are not supported against replica, standby databases.
- High Availability:
  - Oracle GoldenGate Extract does *not* support seamless role transitioning from a primary to a secondary Extract with PostgreSQL high availability configurations. However, manual procedural operations could be followed to provide continuity from the new primary Extract.
  - For more information, see the details available in the Oracle Support note, *Oracle GoldenGate Procedures for PostgreSQL HA Failover (Doc ID 2818379.1)*.

Supported PostgreSQL Data Types

Here's a list of PostgreSQL data types that Oracle GoldenGate supports along with the limitations of this support.

- `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`
- `real`
- `serial`
- `smallint`
- `smallserial`

- text
- time with/without timezone
- timestamp with/without timezone
- uuid
- varchar(n)
- varbit
- xml

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.
- real, double, numeric, 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:

- `Arrays`
- `box`
- `bpchar`
- `circle`
- `Composite Types`
- `Domain Types`
- `line`
- `lseq`
- `Object Identifiers Types`
- `path`
- `pg_lsn`
- `pg_snapshot`
- `point`
- `polygon`
- `Pseudo-Types`
- `Range Types`
- `tsquery`
- `tsvector`
- `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.
- 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.

Topics:

Tables and 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.
- Globalization is supported for object names (table /schema/database/column names) and column data.

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.

Prepare Database Users and Privileges

Learn about required database users, privileges, and permissions for Oracle GoldenGate for SQL Server, including supported SQL Server cloud databases.

Topics:

Oracle GoldenGate 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 GoldenGate processes.

Oracle GoldenGate for SQL Server supports SQL Server authentication for all of its certified platforms and Windows authentication for Classic Architecture only when Oracle GoldenGate is installed on a Windows server.

- To use Windows authentication for Oracle GoldenGate Classic Architecture, the Extract and Replicat processes inherit the login credentials of the Manager process. By default, the Manager process runs interactively as the user logged on to the Windows server or optionally can be added as a Windows Service with a default service name of `GGSMGR`. Whichever method that the Manager process is using, the user that it is running as needs to have the required SQL Server privileges listed above.

- To use SQL Server authentication, create a dedicated SQL Server login for Extract and Replicat and assign the privileges listed below.

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 `GGSCHEMA` 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 4-3 Privileges and Permissions for Oracle GoldenGate User

Privilege	Extract	Replicat	Syntax
msdb Database Roles and Privileges			
`SQLAgentReaderRole`	Yes	No	`ALTER ROLE SQLAgentReaderRole ADD MEMBER gguser;`
`SQLAgentUserRole`	Inherited	Yes	`ALTER ROLE SQLAgentUserRole ADD MEMBER gguser;`
`SELECT ON sysjobactivity`	Yes	No	Required for Classic Architecture only. `GRANT SELECT ON msdb.dbo.sysjobactivity TO gguser;`
`SELECT ON sysjobs`	Yes	No	Required for Classic Architecture only. `GRANT SELECT ON msdb.dbo.sysjobs TO gguser;`
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 gguser;`
			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 msdb.sys.sp_cdc_enable_db 'source_database'`

`DBOWNER`	Yes	Yes	`ALTER ROLE db_owner ADD MEMBER gguser;`

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 `GGSCHEMA` 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 4-4 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 gguser;`
			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 gguser;`
`SQLAgentUserRole`	Inherited	Yes	`ALTER ROLE SQLAgentUserRole ADD MEMBER gguser;`
`SELECT ON sysjobactivity`	Yes	No	Required for Classic Architecture only. `GRANT SELECT ON msdb.dbo.sysjobactivity TO gguser;`
`SELECT ON sysjobs`	Yes	No	Required for Classic Architecture only. `GRANT SELECT ON msdb.dbo.sysjobs TO gguser;`
User Database Roles and Privileges			
`DBOWNER`	Yes	Yes	`ALTER ROLE db_owner ADD MEMBER gguser;`

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 `GGSCHEMA` value used in the `GLOBALS` parameter file.
4. Enable the following privileges and permissions for the Oracle GoldenGate user.

Table 4-5 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 gguser;`

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 `GGSCHEMA` 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.gcloudsql_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 4-6 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, System, and Parameter Settings

Learn about configuring database connection, system, and parameter settings for Oracle GoldenGate for SQL Server.

Topics:

Configuring a Database Connection

Learn about configuring a database connection for SQL Server.

Topics:

Extract and Replicat Database Connectivity

Extract and Replicat connect to a SQL Server database using a system ODBC DSN (Data Source Name) and use ODBC for its metadata queries and transactional data processing.

Creating a Database Connection on Linux

Before creating a database connection for Oracle GoldenGate processes running on Linux, install the latest version of Microsoft ODBC driver for SQL Server (Linux).

Select the following link for download and installation steps:

https://docs.microsoft.com/en-us/sql/connect/odbc/linux-mac/installing-the-microsoft-odbc-driver-for-sql-server?view=sql-server-ver15

For the installation, choose the steps listed under Red Hat Enterprise Linux and Oracle.

After the ODBC software is installed, follow the example below to create an ODBC DSN for Linux:

1. Create a template file for your data source(s):

   ```
   vi odbc_template_file.ini
   ```

2. Describe the data source in the template file. Multiple unique DSN entries can be listed in the template file, if needed.

   In the following example, `mydsn_2019_source` is the DSN name, which will be used with `DBLOGIN` and `SOURCEDB` or `TARGETDB` to connect to the Extract or Replicat to the database.

   ```
   mydsn_2019_source]
   Driver = ODBC Driver 18 for SQL Server
   Server = myserver,1433
   Database = source_database
   TrustServerCertificate=YES
   ```

3. Install the data source using the following command.

   ```
   odbcinst -i -s -f odbc_template_file.ini
   ```

   This command adds the DSN to the system `odbc.ini` file. For more information, select the following link:

   https://docs.microsoft.com/en-us/sql/connect/odbc/linux-mac/connection-string-keywords-and-data-source-names-dsns?view=sql-server-2017

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, depending on the DSN connection used.

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 DSN 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 the DSN. For Linux DSN connections, use the `MultiSubnetFailover=Yes` option in the DSN entry.

Configuring a Database

Learn about configuring a database for SQL Server.

Topics:

SQL Server Supported Versions

Certified versions of SQL Server can be found on the published certification matrix available for each release of Oracle GoldenGate, which is available at the following link:

https://www.oracle.com/middleware/technologies/fusion-certification.html

Oracle GoldenGate Extract supports Enterprise Edition and some versions of SQL Server Standard Edition. Review the Exceptions and Additonal Information column of the certification matrix to see the details of which Standard Edition versions of SQL Server are supported for Extract.

Oracle GoldenGate Delivery 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.

Preparing Tables for Processing

The table attributes in the following sections must be addressed in your Oracle GoldenGate environment.

Topics:

Disabling Triggers and Cascade Constraints on the Target

In an environment where SQL Server is the target, consider triggers and cascade constraints that may repeat an operation that occurred on the source. For example, if the source has an insert trigger on TableA that inserts a record into TableB, and Oracle GoldenGate is configured to capture and deliver both TableA and TableB, the insert trigger on the target table, TableA, must be disabled. Otherwise, Replicat inserts into TableA, and the trigger fires and insert into TableB. Replicat will then try to insert into TableB, and then terminate abnormally.

When a trigger or cascade constraint repeats an operation that occurred on the source, you do not have to disable the trigger or constraint when the following conditions are both true:

- You use the `DBOPTIONS USEREPLICATIONUSER` parameter in Replicat.
- You use OLE DB connection for Replicat. The use of the OLE DB connection is the default configuration. Note that the trigger, constraint, or `IDENTITY` property must have `NOT FOR REPLICATION` enabled.

In the following scenario, disable the triggers and constraints on the target:

- Uni-directional replication where all tables on the source are replicated.

In the following scenarios, enable the triggers and constraints on the target:

- Uni-directional replication where tables affected by a trigger or cascade operation are not replicated, and the only application that loads these tables is using a trigger or cascade operation.
- Uni-directional or -bi-directional replication where all tables on the source are replicated. In this scenario, set the target table cascade constraints and triggers to enable `NOT FOR REPLICATION`, and use the `DBOPTIONS USEREPLICATIONUSER` parameter in Replicat.

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.

Setting the NOT FOR REPLICATION flag for Target Identity Columns, Triggers, and Constraints

1. Set the `NOT FOR REPLICATION` flag on the following objects.
   - Foreign key constraints
   - Check constraints
   - `IDENTITY` columns
   - Triggers (requires textual changes to the definition. See the SQL Server documentation for more information.)

   For active-passive configurations, set it only on the passive database. For active-active configurations, set it on both databases.

2. Partition `IDENTITY` values for bidirectional configurations.
3. In the Replicat `MAP` statements, map the source tables to appropriate targets, and map the child tables that the source tables reference with triggers or foreign-key cascade constraints. Triggered and cascaded child operations are replicated by Oracle GoldenGate, so the referenced tables must be mapped to appropriate targets to preserve data integrity. Make sure to include the same parent and child source tables in the Extract `TABLE` parameters.

> **Note:**
>
> If referenced tables are omitted from the `MAP` statements, no errors alert you to integrity violations, such as if a row gets inserted into a table that contains a foreign key to a non-replicated table.

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.

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.

Transaction Log Settings and Requirements

Know more about transaction log settings and requirements for Oracle GoldenGate for SQL Server.

Topics:

Preparing the Source Database for Oracle GoldenGate Capture

Learn how to enable supplemental logging in the source database and tables that are to be used for capture by the Extract for SQL Server and how to purge older Change Data Capture staging data.

Topics:

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 need to be followed in order to disable or delete the CDC cleanup job immediately after `TRANDATA` is enabled, and to enable Oracle GoldenGate's own CDC cleanup job or Purge Change Data task. See Retaining the CDC Table History 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.gcloudsql_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 `GGSCHEMA 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.

A sample tutorial to setup GoldenGate CDC Capture is available:

Using Oracle GoldenGate for SQL Server CDC Capture ReplicationUsing Oracle GoldenGate 19.1 for SQL Server CDC Capture Replication.

Purging 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 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. Click **Configuration** from the menu on the left to open the **Configuration** page.
2. Click **Tasks** from the **Configuration** page to open the **Tasks** page.
3. Click **Purge Change Data** from the **Tasks** page.
4. 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.

CDC Capture Method Operational Considerations

Learn about the SQL Server CDC Capture options, features, and recommended settings.

Topics:

Tuning SQL Server Change Data Capture

The following information is useful in improving the capture performance of the Extract.

- 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

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, 2. The following versions(s)
  were found: 1. Rerun ADD TRANDATA for all tables previously enabled, including
  heartbeat, heartbeat seed, and filter tables.
  ```

Valid and Invalid Extract Parameters for SQL Server Change Data Capture

This section describes parameters used for the CDC Capture method.

`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 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 Preparing the Source Database for Oracle GoldenGate Capture 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 Preparing the Source Database for Oracle GoldenGate Capture 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.

Topics:

Database Connection

For both Extract and Replicat, it is recommended to create a System DSN 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 DSN settings to point to the new primary.
- For the Extract connecting to the Listener not only allows reconnecting to the primary without editing the DSN 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 DSN 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.
  - Ensure that the Application Intent field of the DSN configuration is set to `READWRITE` and not `READONLY`
  - 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 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`.
- The login used to enable supplemental logging must have `sysadmin` membership of the primary replica instance.
- 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.
- Upon failover from a primary to a secondary replica, reinstall the Oracle GoldenGate CDC Cleanup job on the new primary by re-running the `ogg_cdc_cleanup_setup.bat` file with the `createJob` option.
- 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 Cummulative 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 Chang 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:

- 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.

Topics:

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 schemas or objects are not be automatically replicated by Oracle GoldenGate unless they are explicitly specified without a wildcard.

- `"sys"`
- `"cdc"`
- `"INFORMATION_SCHEMA"`
- `"guest"`

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

Add Database Credentials

You must have a working database credential for your Extract and Replicat processes.

1. Launch the Administration Service interface and log in.
2. Click **Configuration** from the **Application Navigation** pane.
3. Click the plus sign (**+**) sign next to Credentials.
4. Enter the following details in the displayed fields:

Database Credential Options	Description
Credential Domain	Specify a domain name to which the database credential is associated. For example, "OracleGoldenGate" is the default domain name, incase you don't specify a domain name.
Credential Alias	This is the alias for your database credential.
User ID	This is the username of the database user. For Oracle database, if you use the EZconnect syntax to connect to the database, then you can specify the value in this field in the following manner: *dbusername*`@`*hostname*`:port/service_name` `dbusername` is the database user name. `hostname` or IP address of the server where the database is running. `port` is the port number for connecting to the database server. Usually, this value is 1521. `service_name` is the name of the service provided in the tnsnames.ora file for the database connection.
Password	Password used by database user to log in to the database.

5. Click **Submit**.
6. Click the Connect to database icon to test that the connection is working correctly. If the connection is successful, the Connect to database icon turns blue. You'll also see sections to set up checkpoint and heartbeat tables after the connection is successful.

Before Adding Extract and Replicat Processes

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

Access the Configurations Page

Configure connections to the database from Oracle GoldenGate by setting up database user credentials from the Configurations page of the Administration Service left-navigation pane.

See Add Database Credentials for steps to create credentials for the database and test the connection. You can set up database credentials to set up connections to multiple databases, as required by the Extract and Replicat processes.

Enable TRANDATA

Valid for Oracle and Non-Oracle databases.

Depending on the source database, supplemental logging must be enabled. This can be done at the table, schema, or global (database) level.

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

Oracle: Enable TRANDATA or SCHEMATRANDATA

Valid for Oracle.

Depending on the source database, supplemental logging must be enabled. This can be done at the table, schema, or global (database) level.

To enable supplemental logging at the table and schema level, on **Configuration** page:

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, schema.table1. You can also use wildcard instead of specific table name.
3. Select the `Add TRANDATA` Information in the background? option as required.
4. Click **Submit**.

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

Db2 z/OS: Enable Change Capture

Follow these steps to 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 with update statements.

1. From the Oracle GoldenGate directory, start the Admin Client.
2. Log on to Db2 as a user that has `ALTER TABLE` privileges.

   ```
   DBLOGIN SOURCEDB DSN, USERID user[, PASSWORD password][,
   encryption_options]
   ```

3. Issue the following command. where *`table`* is the fully qualified name of the table. You can use a wildcard to specify multiple table names but not owner names.

   ```
   ADD TRANDATA table
   ```

   By default, `ADD TRANDATA` issues the following command:

   ```
   ALTER TABLE name DATA CAPTURE CHANGES;
   ```

SQL Server: Enable Supplemental Logging and Other Features

A database user must issue the `ADD TRANDATA` command to enable supplemental logging on the source database in an Oracle GoldenGate configuration. A database login command (`DBLOGIN`) is issued from the command line interface before `ADD TRANDATA` is issued.

- The database user that enables `TRANDATA` must have `sysadmin` rights.

Extract can run with dbowner permissions. However, you also need `sysadmin` rights to issue the `ADD/ALTER/ DELETE/INFO HEARTBEATTABLE` commands, or to create the Oracle GoldenGate CDC Cleanup job using the `ogg_cdc_cleanup_setup.bat` batch file.

Add a Checkpoint Table

A checkpoint table is required for all non-parallel Replicats and must be created in the database prior to adding a Replicat. You can view the checkpoint table from the Checkpoint section of the **Configurations** page.

To add a checkpoint table:

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

   `table.checkpoint_table_name`

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

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

Add Heartbeat Table

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

To create the heartbeat table, you have to follow these steps on the source and target system:

> **Note:**
>
> Creating the heartbeat table is optional but is recommended.

1. From the Administration Service, select Configuration from the navigation pane.
2. Select the + sign next to the Heartbeat section of the Database tab. You'll need to enter the values for the heartbeat frequency, retention time, and purge frequency.

Here are the steps to add a heartbeat table from the Admin Client:

1. Launch the Admin Client from the command line.
2. Connect to the deployment from the Admin Client.

   ```
   CONNECT https://remotehost:srvmgrport DEPLOYMENT deployment_name AS
   deployment_user PASSWORD deployment_password
   ```

   Here's an example:

   ```
   CONNECT https://remotehost:16000 DEPLOYMENT ggdep_postgres AS ggadmin
   PASSWORD P@ssWord
   ```

3. Connect to the source and target databases using the `DBLOGIN USERIDALIAS` command. The following example shows the connection to the source database with credential alias `ggeast`:

   ```
   (https://remotehost:16000 ggdep_postgres)> DBLOGIN USERIDALIAS ggeast
   ```

4. Add the heartbeat table:

   ```
   (https://remotehost:16000 ggdep_postgres)> ADD HEARTBEATTABLE
   ```

Optionally, for a target only database, one that is used for unidirectional replication only, you can include the `TARGETONLY` option which will not create a heartbeat record update function.

See `ADD HEARTBEATTABLE` for details about command options.

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.

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.

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.

5
Quickstarts

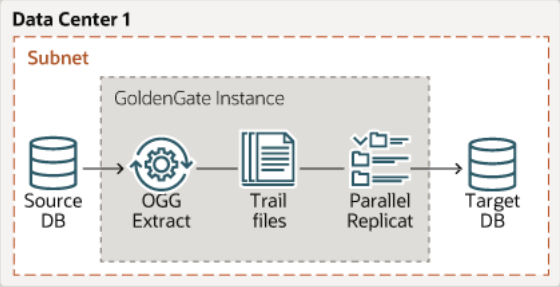
This section lists the Oracle GoldenGate quickstarts.

Topics:

Set Up Data Replication with Oracle GoldenGate Microservices Architecture

Use this quickstart to configure data replication using Oracle GoldenGate Microservices Architecture for a multitenant container database with two pluggable databases to demonstrate data replication from an Oracle to Oracle database in a HUB configuration.

> **Note:**
>
> This quickstart does not perform an initial load instantiation and assumes that the tables and data are the same in the source and target endpoints.



The source and target databases in this diagram refer to the container and pluggable databases (PDBs).

Container Database (CDB$ROOT) Process Names	Pluggable Database (DBEAST) Process Names	Pluggable Database (DBWEST) Process Name
• **CDB$ROOT database user: c##ggadmin** • **Database credential alias: cggnorth**	• Database user: **ggadmin** • Database alias: **ggeast** • Extract: **exte**	• Database user: **ggadmin** • Database alias: **ggwest** • Extract: **extw**

Configure and Set Privileges for Oracle Multitenant Database

In Oracle database, you need to enable replication for Oracle GoldenGate and assign privileges to the database user at the CDB level and the pluggable database (PDB) level.

The database is in `ARCHIVELOG` mode and `FORCE LOGGING` and Supplemental Logging is enabled. For the container database, assign the following privileges to the common user (`cdb$root`):

CDB User Privileges

```
## CGGNORTH DATABASE SETUP AT CDB LEVEL
ALTER SESSION SET CONTAINER=cdb$root;
ALTER SYSTEM SET ENABLE_GOLDENGATE_REPLICATION=TRUE;
ALTER SYSTEM SET STREAMS_POOL_SIZE=2G;
ALTER DATABASE FORCE LOGGING;
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
ARCHIVE LOG LIST;
CREATE TABLESPACE GG_DATA DATAFILE '+DATA' SIZE 100M AUTOEXTEND ON NEXT 100M;
CREATE USER c##ggadmin IDENTIFIED BY PASSWORD CONTAINER=ALL DEFAULT
TABLESPACE GG_DATA TEMPORARY TABLESPACE TEMP;
GRANT ALTER SYSTEM TO c##ggadmin CONTAINER=ALL;
GRANT DBA TO c##ggadmin CONTAINER=ALL;
GRANT CREATE SESSION TO c##ggadmin CONTAINER=ALL;
GRANT ALTER ANY TABLE TO c##ggadmin CONTAINER=ALL;
GRANT RESOURCE TO c##ggadmin CONTAINER=ALL;
EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE(' c##ggadmin
',CONTAINER=>'ALL');
```

Source PDB User Privileges (`DBEAST`)

```
ALTER SESSION SET CONTAINER=dbeast;
CREATE TABLESPACE GG_DATA DATAFILE '+DATA' SIZE 100M AUTOEXTEND ON NEXT 100M;
CREATE USER ggadmin IDENTIFIED BY PASSWORD CONTAINER=CURRENT;
GRANT CREATE SESSION TO ggadmin CONTAINER=CURRENT;
GRANT ALTER ANY TABLE TO ggadmin CONTAINER=CURRENT;
GRANT RESOURCE TO ggadmin CONTAINER=CURRENT;
GRANT DBA TO ggadmin CONTAINER=CURRENT;
EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE('ggadmin');
```

Target PDB User Privileges (`DBWEST`):

```
ALTER SESSION SET CONTAINER=dbwest;
CREATE USER ggadmin IDENTIFIED BY PASSWORD CONTAINER=CURRENT;
GRANT ALTER SYSTEM TO ggadmin CONTAINER=CURRENT;
GRANT CREATE SESSION TO ggadmin CONTAINER=CURRENT;
GRANT ALTER ANY TABLE TO ggadmin CONTAINER=CURRENT;
GRANT RESOURCE TO ggadmin CONTAINER=CURRENT;
GRANT DBA TO ggadmin CONTAINER=CURRENT;
GRANT DV_GOLDENGATE_ADMIN, DV_GOLDENGATE_REDO_ACCESS 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. In this quickstart, the assumption is that the database user is a database administrator. See Grant User Privileges for Oracle Database 21c and Lower and Configure a Multitenant Container Database to know more about specific privilege requirements.

Configure the Replication Process from Oracle GoldenGate MA Web Interface

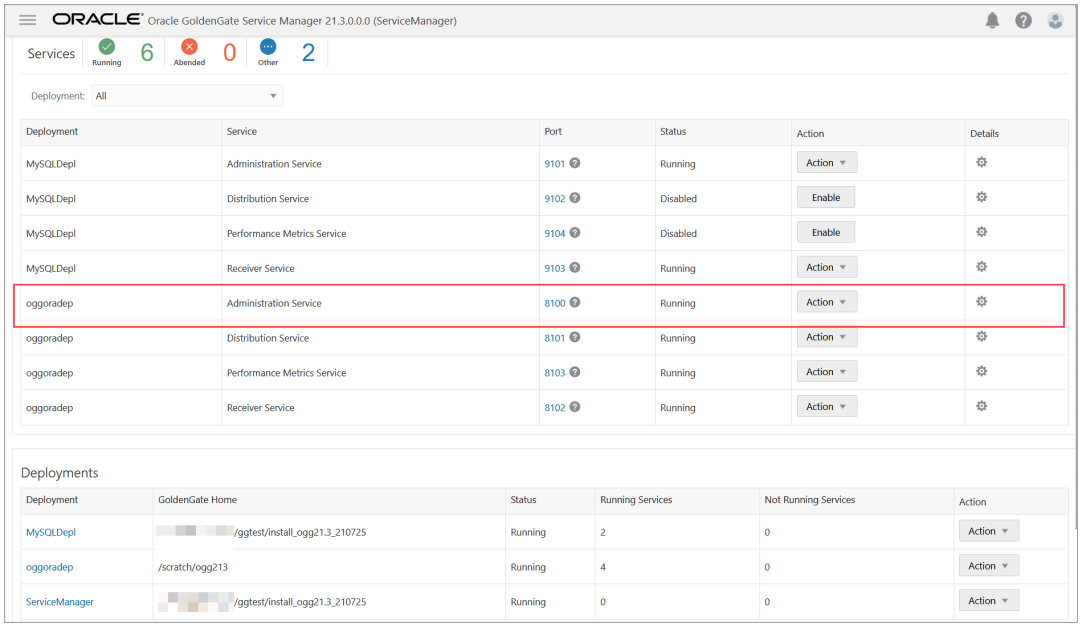
Data replication processes include Extracts, Replicats processes, along with Distribution Paths (`DISTPATH`) or Receiver Paths or Target-Initiated Paths (`RECVPATH`).

Using the following steps, you'll be able to configure data capture (Extract) and apply (Replicat) processes. You'll also be able to test if the replication has started. The `DISTPATH` process is not used for this configuration.

Step 1: Add Database Credentials from the Administration Service

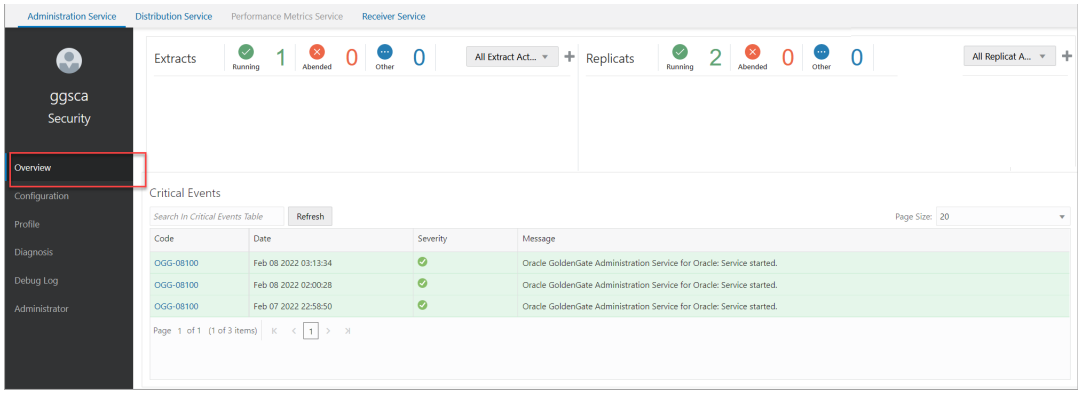
In this section, you'll add the database credentials to connect to the source and target databases using EZConnect.

1. Keep your database user credentials, which created in the previous session, ready. You'll use them to connect Oracle GoldenGate to the database server.
2. Open the Service Manager login page in a web browser and log in to the Service Manager with your Oracle GoldenGate administrator user credentials. If logging in for the first time, you have to log in with the administrator account user credentials, created when adding your deployment with Oracle GoldenGate Configuration Assistant wizard.
3. From the Service Manager Overview page, click the port number for the Administration Service of the deployment.

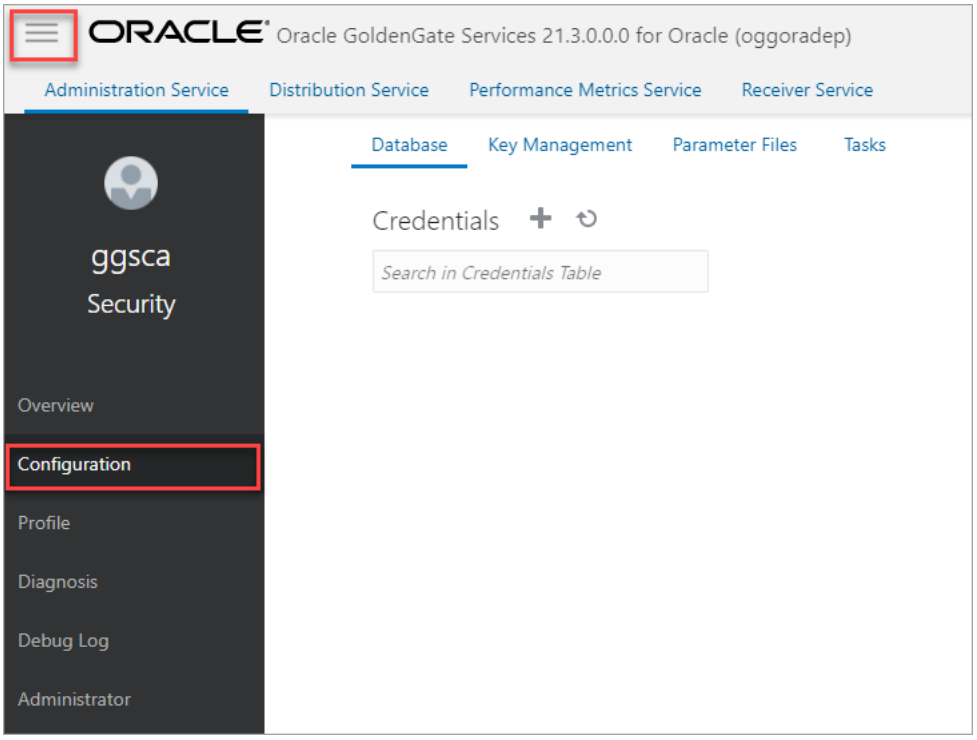


This opens the Administration Service login page.

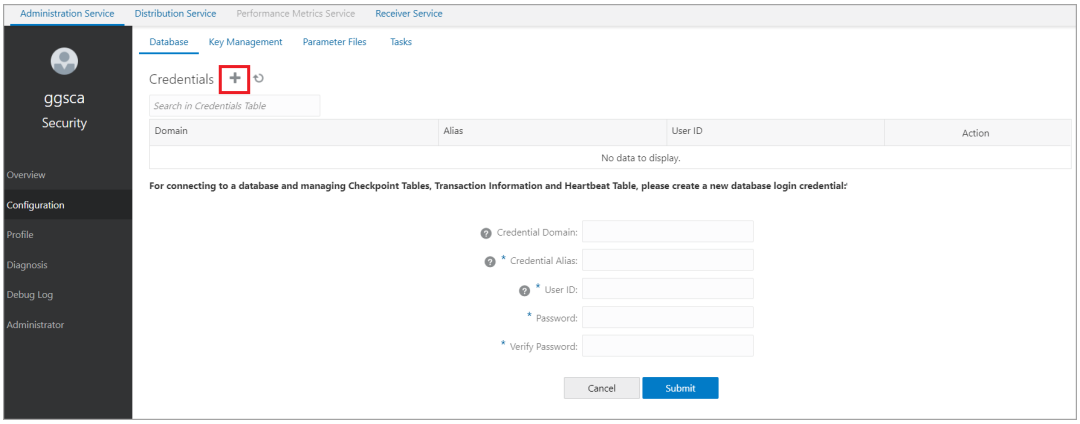
4. Log in to the Administration Service using the same credentials, which you used to log in to the Service Manager. The Administration Service **Overview** page is displayed.



5. Click the **Application Navigation** icon to open the left-navigation pane and click **Configuration** to open the **Database** tab of the **Configuration** page.



6. Click the plus (+) sign in the **Credentials** section to begin adding database user credentials.



7. You need to add connections for container database (CDB) and pluggable databases (PDBs). Each CDB is used to capture (Extract) from the source database and PDB for delivery (Replicat).

   Use the EZconnect syntax to configure the database connection. You need the username, password, hostname, port number, and service name connection information to use the EZConnect syntax.

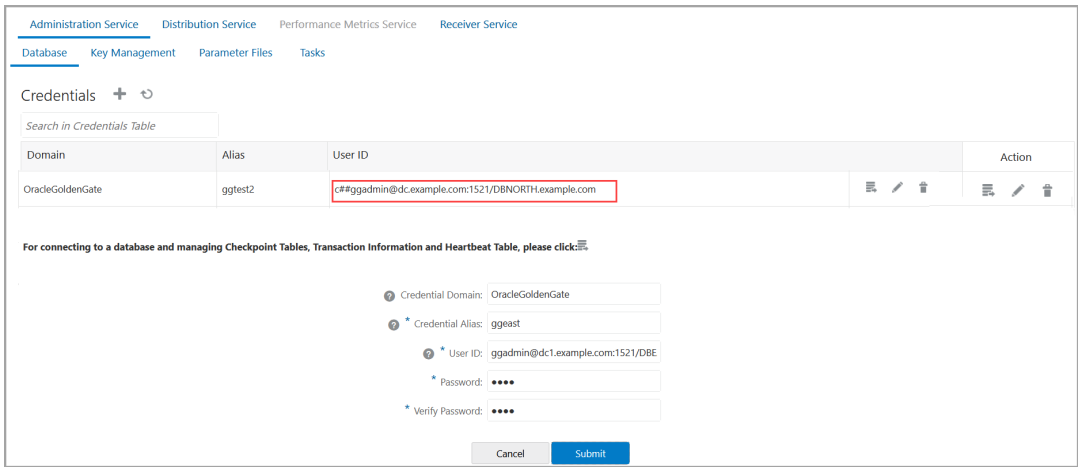
   Here's the syntax that you need to specify in the User ID field:

   username@hostname:port/service_name

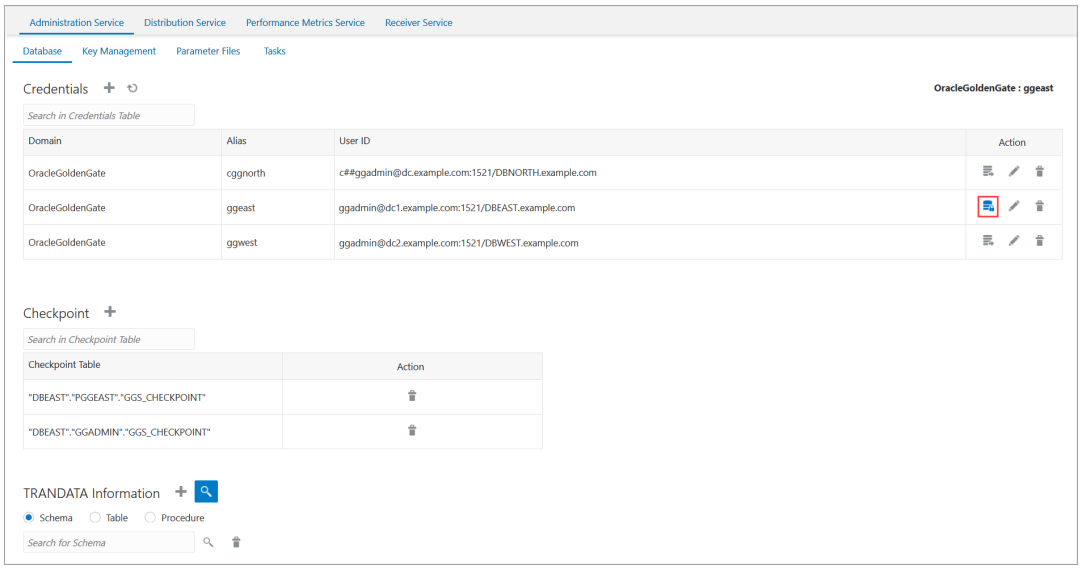
   Here's an example for setting the User ID with EZConnect:

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

The following screen shows the database credential (**cggnorth**) for connecting to the user `c#ggadmin` added to the credentials list. You can also see the credentials being added for the **Alias** (`ggeast`) for connecting to the `DBEAST` PDB.



8. Click on the blue icon in the **Actions** column to connect to the database. The icon turns blue when the connection is successful.

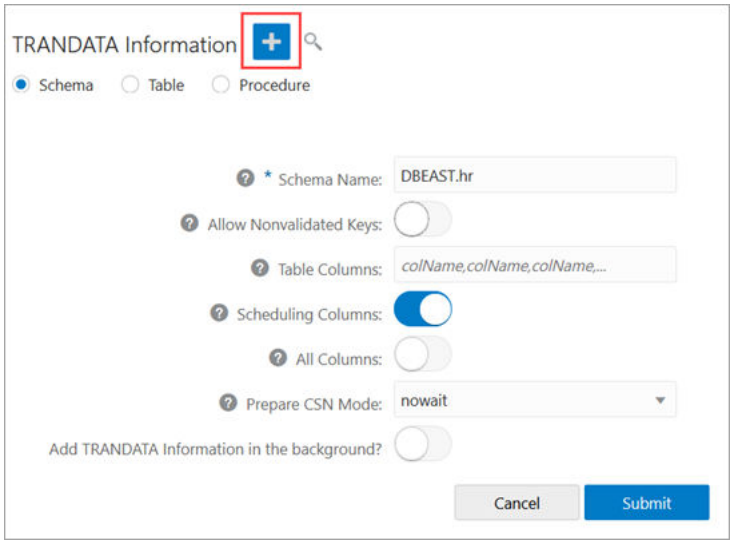


After connecting to the database, the sections to add checkpoint table, TRANDATA, and heartbeat table are displayed.

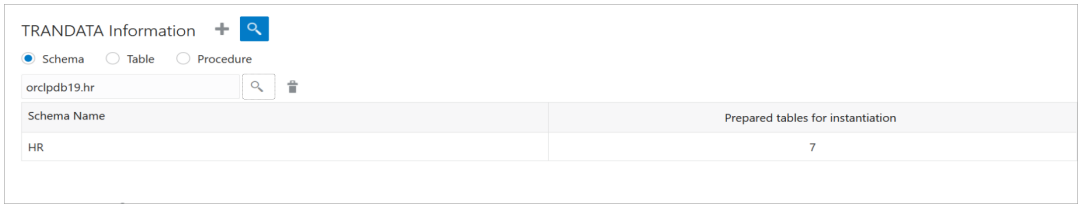
Step 2: Add TRANDATA, Heartbeat, and Checkpoint Tables

In this section, you will add TRANDATA for the source database to enable writing information to the redo logs. This would ensure that the rows added to the source database are uniquely identified on the target database and are updated. You'll add heartbeat tables for the source and target databases to monitor any possible lags. You will also add a checkpoint table for the target database to ensure that if there is a failure, then the Extract and Replicat processes can restart from the point of failure.

1. Add TRANDATA to the source connection. Use the **TRANDATA Information** section to set up database logging properties. This is an essential step to enable supplemental logging and ensure that the data is written to the database redo log.

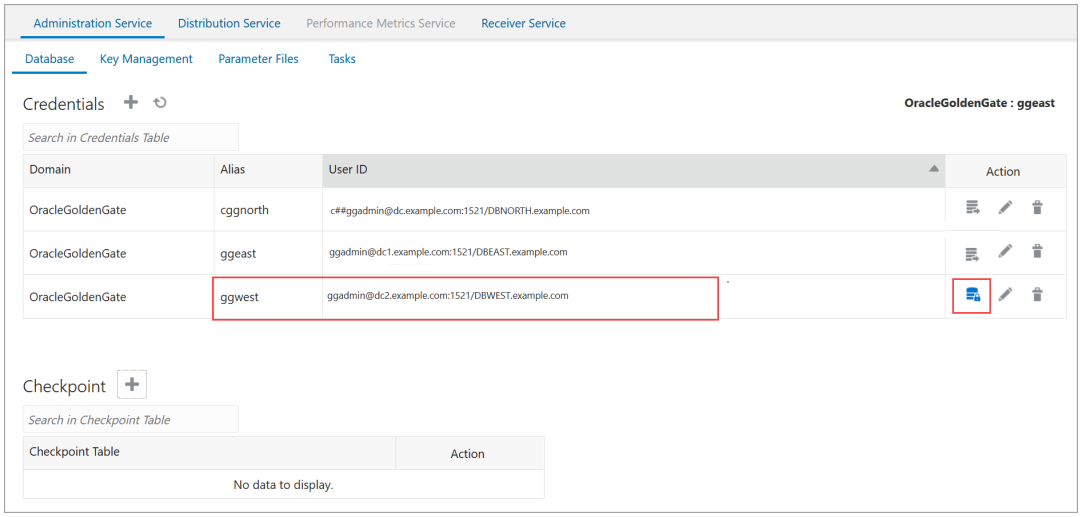


After you add the trandata, you can search for the schema for which you've add the trandata, using the search icon. This will display the trandata information. The following image shows the trandata information for the `HR` schema in the pluggable database `DBEAST`.

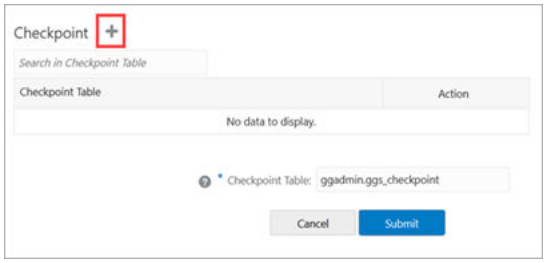


See Configure Logging Properties to learn the steps for configuring the logging properties at the Schema, Table or Procedure level.

2. To set up the checkpoint table for Replicat, you need to connect to the target database credentials (**ggwest**) from the Credentials section.



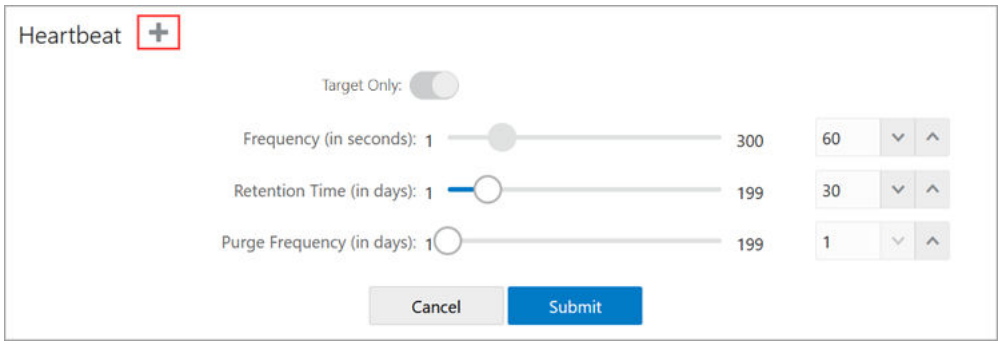
3. Click the plus sign (+) to add the checkpoint table for the target (pluggable) database.



Click **Submit**. The checkpoint table is added.

Also see the Before Adding Extract and Replicat Processes section, for details on creating heartbeat tables.

4. Add the heartbeat tables for both source and target endpoints by connecting to **ggeast** and **ggwest** database credential aliases. Add the heartbeat table by clicking the plus sign.

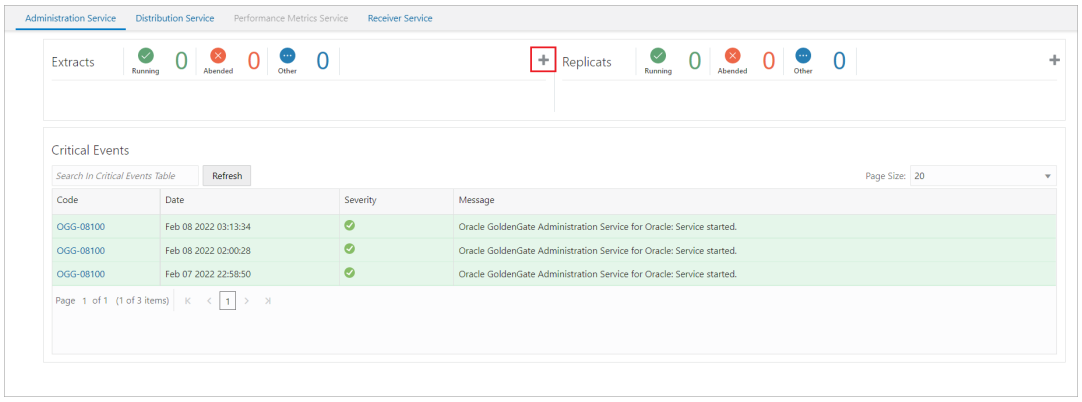


5. Click **Submit** after adjusting the heartbeat options.

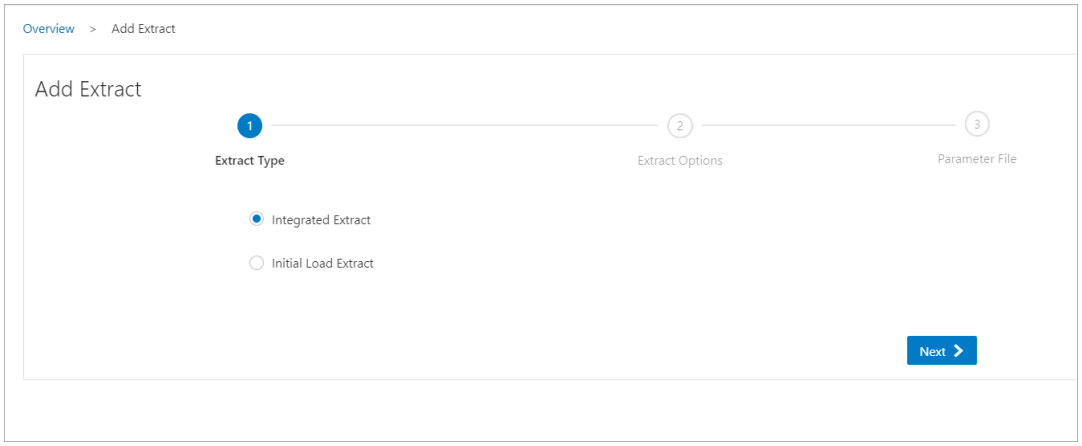
Step 3: Add an Extract

In this section, you will add an Extract process (**exte**). The Extract process captures data from the source database and writes it to a trail file (**ea**).

1. Click the **Overview** option from the left-navigation pane of the **Administration Service** and click the plus sign (+) from the Extract section.

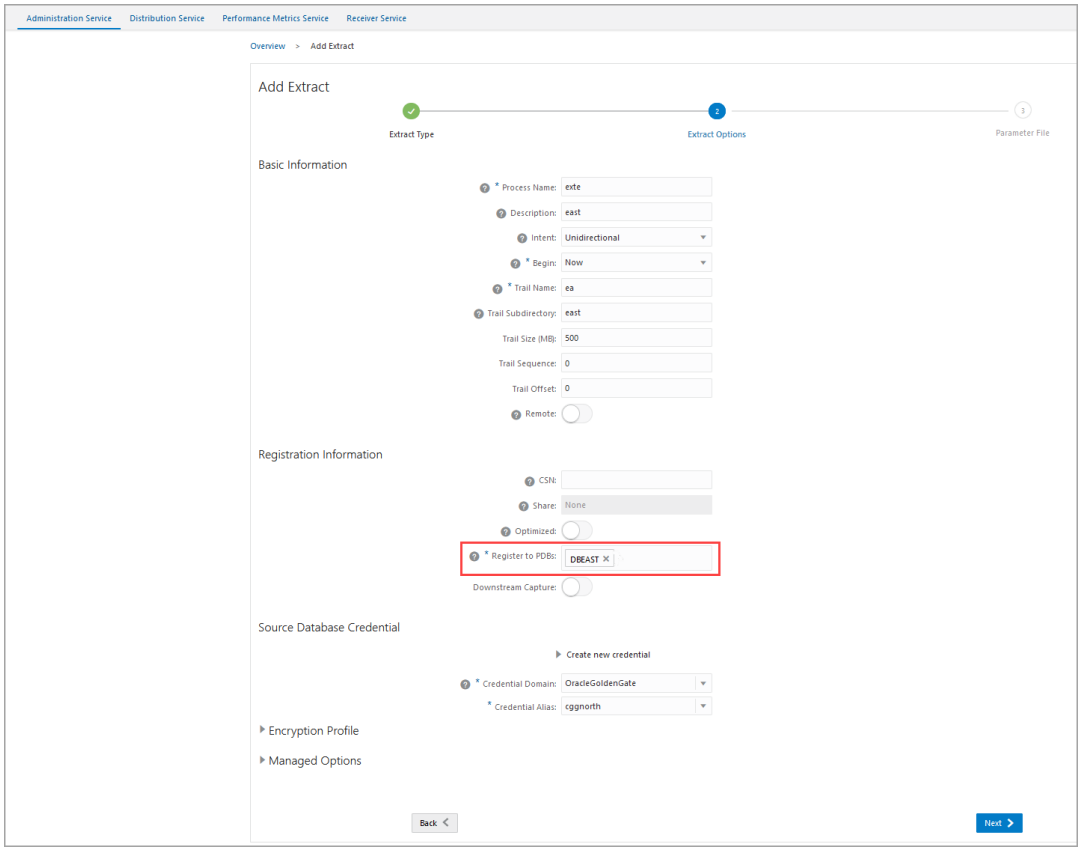


2. From Add Extract wizard, select **Integrated Extract**.



> **Note:**
>
> Before creating Replicat, you need to create an initial load Extract when starting the replication process for the first time. To learn about the initial load Extract and it's use case, see Add Initial Load Extract Using the Admin Client.

3. Click **Next** and specify the Extract options in the Extract Options screen. See the detailed steps to add an Extract from the Add a Primary Extract section.



If you are creating the Extract for a pluggable database, then you'll see option **Register to PDBs** as soon as you enter the credentials domain and alias. Select the PDB in the container database that you want to use for replication.

4. After you enter the options for the Extract, click **Next**. The next screen displays the Extract parameter file to help you review the Extract settings.

   Here's the Extract parameter file for the Extract exte:

   ```
   EXTRACT exte
   USERIDALIAS cggnorth DOMAIN OracleGoldenGate
   EXTTRAIL east/ea
   SOURCECATALOG DBEAST
   DDL INCLUDE MAPPED
   TABLE hr.*;
   ```

   Review these settings and update the Extract configuration as needed.

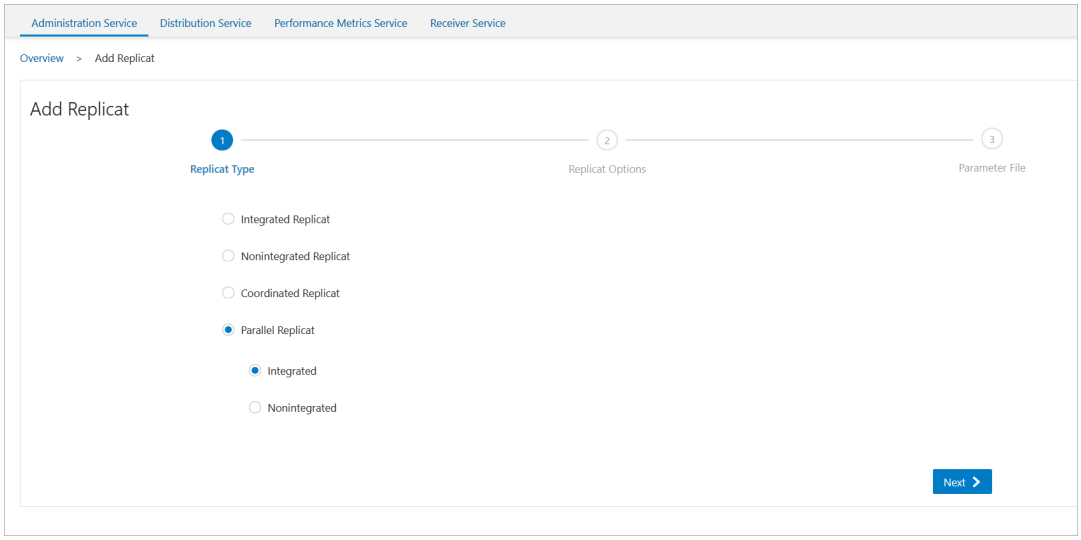
   For multitenant databases, you need to add entries for Extract to capture from multiple pluggable databases to a single trail. In the parameter file, source objects must be specified in `TABLE` and `SEQUENCE` statements with their fully qualified three-part names in the format of *`container.schema.object`* or using the `SOURCECATALOG` parameter with two-part names *`schema.object`*.

5. Click **Create and Run** to start your Extract.

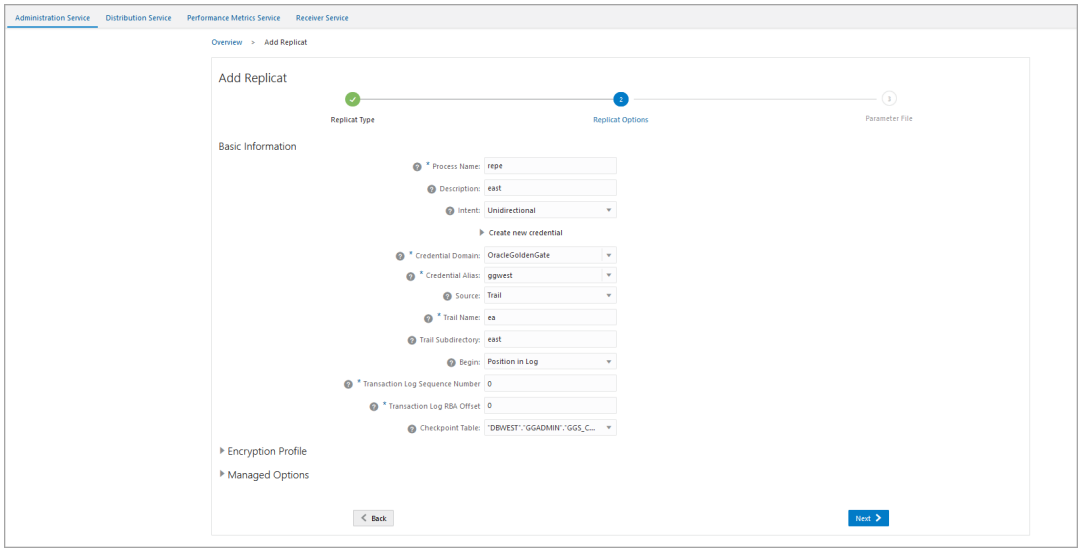
Step 4: Add a Replicat

In this section, you will add a Replicat process (**repe**). The Replicat process delivers the change data from the trail file (**ea**) created by the Extract, to the target database. Replicat reads the trail file on the target database, reconstructs the DML or DDL operations, and applies them to the target database.

1. Before you Add a Replicat, make sure that you added your checkpoint table for the target database (DBWEST) by connecting to the **ggwest** database credentials.
2. Select a Replicat type to deliver data to the target database. Follow the wizard to complete adding a Replicat. See Add a Replicat.



3. Enter the **Parallel Nonintegrated Replicat** options in the Replicat Options screen.



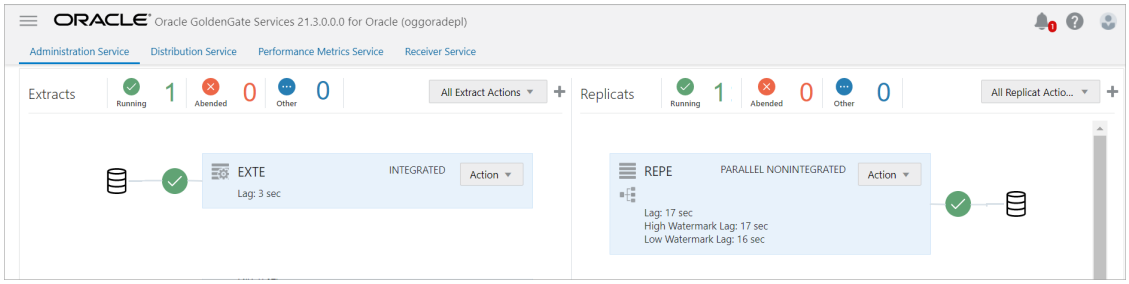
4. Click **Next** to view the **Replicat Parameter File** screen. All the parameters that you have specified are available for review here.

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

Here's a sample of the Replicat Parameter File:

```
REPLICAT repe
USERIDALIAS ggwest DOMAIN OracleGoldenGate
--DDL EXCLUDE ALL
DDLERROR default discard
REPERROR (default,discard)
DDLOPTIONS REPORT
SOURCECATALOG DBEAST
MAP hr.*, TARGET hr.*;
```

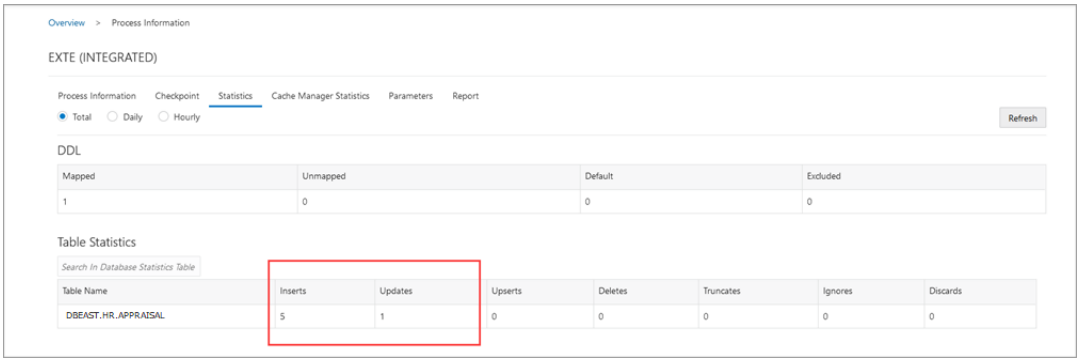
After the Replicat starts successfully, you can see the Extract and Replicat processes in running state on the **Administration Service** Overview page.



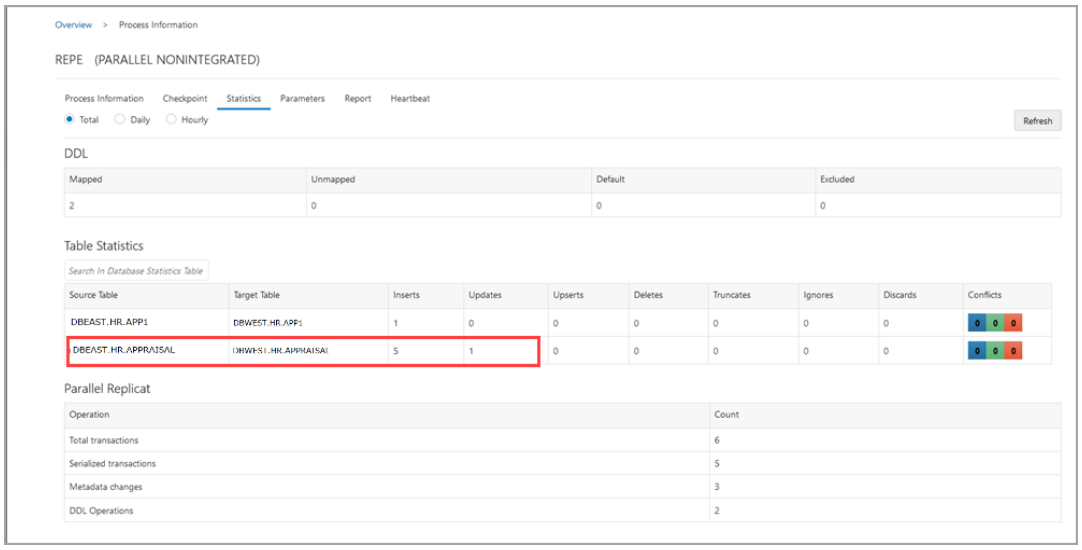
Step 5: Test the Replication

To test if the replication has started, try insert, update, or delete operations on your database and then follow these steps:

1. Click **Action** from the Extract (**exte**) section and click **Details**.
2. Click the **Statistics** tab. You'll see additions to the **Insert**, **Updates**, or **Deletes** columns on this page.

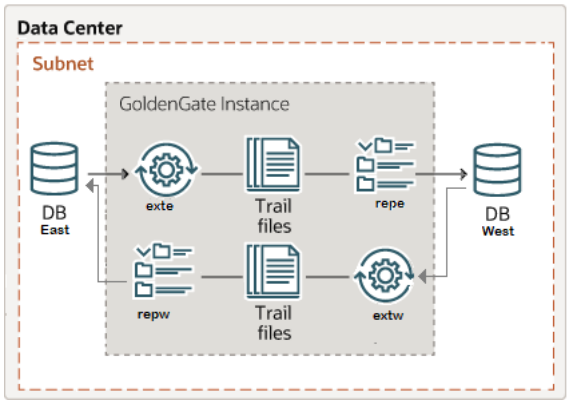


Also see the **Statistics** tab using the Replicat **Details** option. You would see the updates in the **Table Statistics** section.



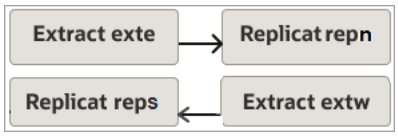
Set Up Bidirectional Replication for Oracle GoldenGate Microservices Architecture

This quickstart demonstrates an active-active bidirectional replication between two pluggable databases over a single multitenant container Oracle database instance.



An active-active bidirectional replication implies that both data sources and targets (PDBs in this case), have the potential to send updates to each other. There are two data sources with identical sets of data that can be changed by application users on either side. Oracle GoldenGate replicates transactional data changes from each database to the other to keep both sets of data current.

The following diagram depicts the bidirectional replication workflow shown in this quickstart:



> **Note:**
>
> This quickstart uses a single multitenant container database with two PDBs to demonstrate bidirectional replication between two PDBs. However, in most real-life scenarios, bidirectional data replication happens across different multitenant container databases or different database instances.

Here's what's covered:

- Process Names in the Bidirectional Data Replication Environment
- Considerations for Configuring a Bidirectional Replication
- Oracle Multitenant Container Database Privileges Required for Data Replication
- Automatic Conflict Detection and Resolution (ACDR) Configuration
- Bidirectional Data Replication Process Configuration

Process Names in the Bidirectional Data Replication Environment

The following nomenclature is used to refer to processes for the database and Oracle GoldenGate.

Container Database (CDB$ROOT) Process Names	Pluggable Database (DBEAST) Process Names	Pluggable Database (DBWEST) Process Name
• **CDB$ROOT database user: c##ggadmin** • **Database credential alias: cggnorth**	• Database user: **ggadmin** • Database alias: **ggeast** • Extract: **exte** • Replicat **repn**	• Database user: **ggadmin** • Database alias: **ggwest** • Extract: **extw** • Replicat: **reps**

On DBWest:

Considerations for Configuring a Bidirectional Replication

To maintain data integrity and avoid conflicts, you need to configure the Extract and Replicat processes to prevent data looping and conflict using certain parameters and the automatic conflict detection and resolution (ACDR) feature.

Ideally, all situations that could lead to potential conflicts in a bidirectional or multidirectional replication must be avoided. However, if conflicts occur, Oracle GoldenGate provides the automatic conflict detection and resoution (ACDR) feature to handle them.

- **At the PDB level:**

  The Automatic Conflict Detection and Resolution feature (ACDR) available with Oracle database, allows you to manage conflict detection and resolution using the `DBMS_GOLDENGATE_ADM` package, using the `ADD_AUTO_CDR` procedure. You need to enable this package at the database level on both PDBs in this case. See Enable ACDR.

- **Oracle GoldenGate Extract parameter settings**

  – `LOGALLSUPCOLS`: This parameter controls writing of supplementally logged columns specified using `ADD TRANDATA` and the columns enabled for Conflict Detection and Resolution (CDR) in Oracle GoldenGate. This parameter is set by default for Extract.

  – `UPDATERECORDFORMAT`: This parameter is set by default for integrated Extract, so don't need to set it in the parameter file. Its function is to combine the before and after images of an `UPDATE` operation into a single record in the trail. The `COMPACT` option

generates one trail record that contains the before and after images of an `UPDATE`, where the before image includes all 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`.

  - `EXCLUDETAG` option ensures that there is not looping of data. Looping of data happens when a database sends updates to the second database and the second database assumes those updates to be a new changes, and tries to replicate this update back to the source database itself. These parameter settings are done when configuring the Extract parameter file, as shown in Step 3: Add Extracts of this document.

- **Oracle GoldenGate Replicat parameter settings**:

  ACDR works with integrated Replicat or parallel integrated Replicat. See the Replicat Parameter file in this document to know more.

Set the Required Privileges for Oracle Multitenant Database

In Oracle database, you need to enable replication for Oracle GoldenGate and assign privileges to the database user at the CDB level and the pluggable database (PDB) level.

The database is in `ARCHIVELOG` mode and `FORCE LOGGING` and Supplemental Logging is enabled. For the container database, assign the following privileges to the common user (`cdb$root`):

CDB User Privileges

```
## CGGNORTH DATABASE SETUP AT CDB LEVEL
ALTER SESSION SET CONTAINER=cdb$root;
ALTER SYSTEM SET ENABLE_GOLDENGATE_REPLICATION=TRUE;
ALTER SYSTEM SET STREAMS_POOL_SIZE=2G;
ALTER DATABASE FORCE LOGGING;
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
CREATE USER c##ggadmin IDENTIFIED BY PASSWORD CONTAINER=ALL DEFAULT
TABLESPACE GG_DATA TEMPORARY TABLESPACE TEMP;
GRANT CONNECT, RESOURCE, DBA TO c##ggadmin CONTAINER=ALL;
GRANT CREATE SESSION TO c##ggadmin CONTAINER=ALL;
EXEC
DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE('c##ggadmin',CONTAINER=>'ALL');
```

PDB User Privileges for DBEAST

```
ALTER SESSION SET CONTAINER=dbeast;
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');
```

PDB User Privileges for `DBWEST`

```
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. In this quickstart, the assumption is that the database user is a database administrator. See Grant User Privileges for Oracle Database 21c and Lower and Configure a Multitenant Container Database to know more about specific privilege requirements.

Enable ACDR

Before enabling ACDR at the database level, it is recommended that you stop any running Extract or Replicat processes. To enable ACDR for the PDB `DBEAST`:

```
EXEC DBMS_GOLDENGATE_ADM.ADD_AUTO_CDR('hr', 'employees',
RECORD_CONFLICTS=>TRUE);
```

The output will show as:

```
PL/SQL procedure successfully completed.
```

Now, switch to the other PDB, `DBWEST` and run the same command:

```
EXEC DBMS_GOLDENGATE_ADM.ADD_AUTO_CDR('hr', 'employees');
```

This enables the ACDR package on both PDBs.

You can check if ACDR has been enabled for the PDBs by checking for invisible columns that are added to manage ACDR at the column level. Run the following commands to test this:

Use the view `ALL_GG_AUTO_CDR_TABLES` to list down the columns used for ACDR in the PDBs:

```
SELECT table_owner, table_name, tombstone_table, row_resolution_column, FROM
all_gg_auto_cdr_tables;
```

The output for this command shows:

```
TABLE_OWNER
--------------------------------------------------------------------------------
--
TABLE_NAME
--------------------------------------------------------------------------------
--
TOMBSTONE_TABLE
--------------------------------------------------------------------------------
--
ROW_RESOLUTION_COLUMN
--------------------------------------------------------------------------------
--
```

```
HR
EMPLOYEES
DT$_EMPLOYEES
CDRTS$ROW
```

Notice the two invisible columns that are added here:

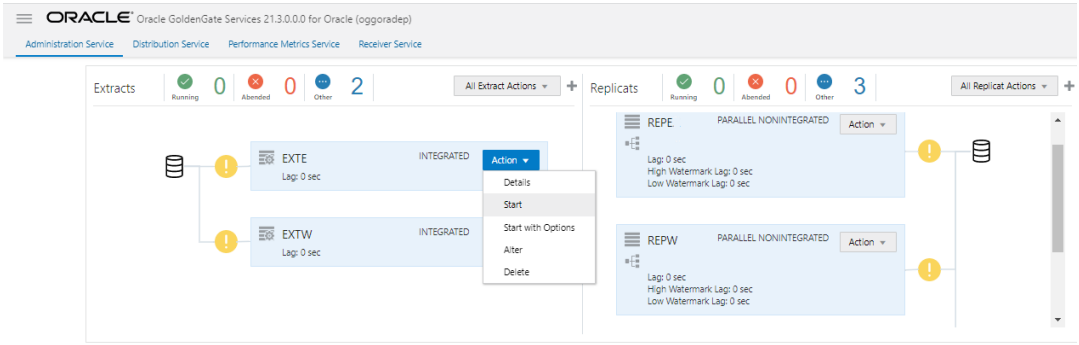
- `DT$_EMPLOYEES`: This is the tombstone table used for locking any delete transactions.
- `CDRTS$ROW`: This is the row resolution column. When there is a conflict, this column which contains the timestamp for the transaction, is used to decide the record that would be applied in a row. This implies that the record with the latest timestamp would be used to apply changes in the row.

These columns are appended to *`schema.table`* on both PDBs, `DBEAST` and `DBWEST`.

After you have enabled ACDR, you'll need to edit the Replicat parameter file to include the invisible columns. Add the `MAPINVISIBLECOLUMNS` parameter in the Replicat parameter file, to allow Replicat to include target columns with default column mapping. This is explained in detail when configuring the Replicat Parameter File in **Step 4: Add Replicat** section.

Restart the Extract and Replicat processes from the web interface:

1. Log in to the Administration Service web interface.
2. From the Administration Service Overview page, click the **Action** button next to the Extract process, **exte**.
3. Click **Start**.



   The green check mark would appear next to the process indicating that the processes started successfully.

Similarly, start the other Extract and Replicat processes on both PDBs.

Configure the Replication Process from Oracle GoldenGate MA Web Interface

Using the following steps, you'll be able to configure data capture (Extract) and apply (Replicat) processes. You'll also be able to test if the replication has started.

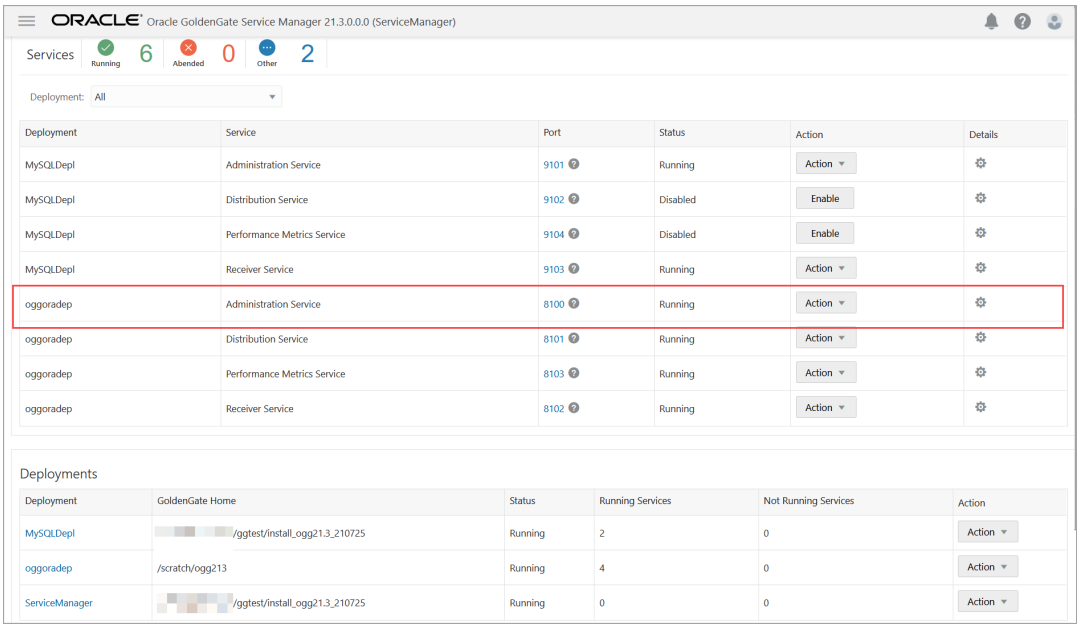
- Step 1: Add Database Credentials from the Administration Service
- Step 2: Add Heartbeat and Checkpoint Tables
- Step 3: Add Extracts
- Step 4: Add a Replicat
- Test and Monitor Transactions
- Test Automatic Conflict Detection and Resolution

The `DISTPATH` process is not used for this configuration.

Step 1: Add Database Credentials from the Administration Service

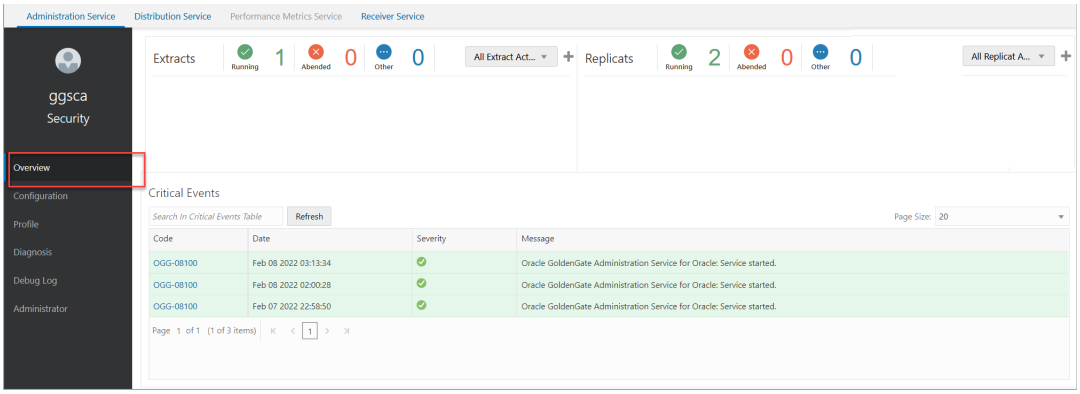
In this section, you'll add the database credentials to connect to the source and target databases using EZConnect.

1. Keep your database user credentials, which created in the previous session, ready. You'll use them to connect Oracle GoldenGate to the database server.
2. Open the Service Manager login page in a web browser and log in to the Service Manager with your Oracle GoldenGate administrator user credentials. If logging in for the first time, you have to log in with the administrator account user credentials, created when adding your deployment with Oracle GoldenGate Configuration Assistant wizard.
3. From the Service Manager Overview page, click the port number for the Administration Service of the deployment.

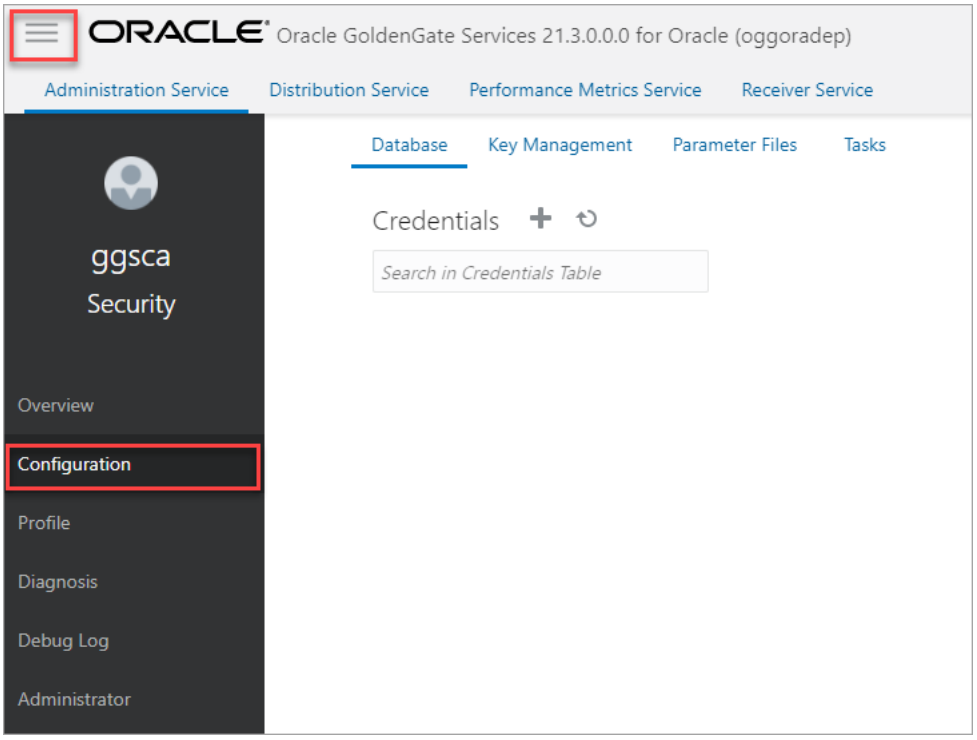


This opens the Administration Service login page.

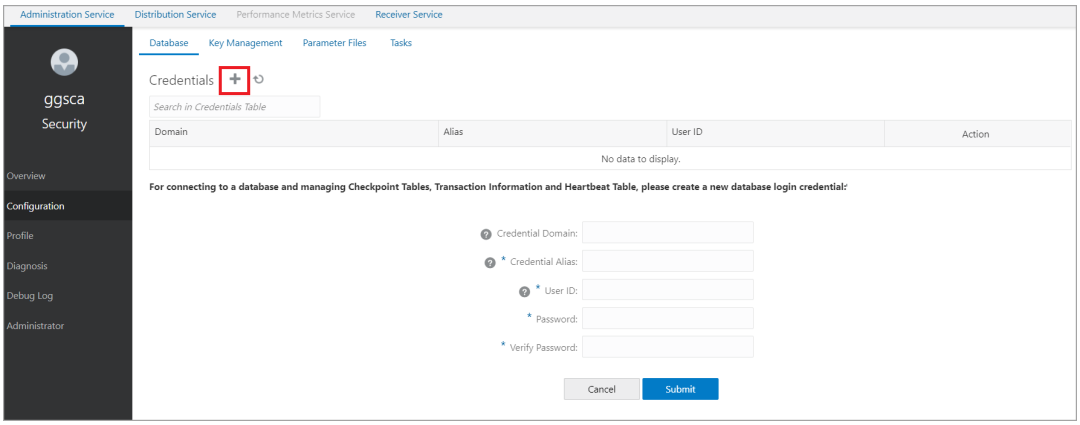
4. Log in to the Administration Service using the same credentials, which you used to log in to the Service Manager. The Administration Service **Overview** page is displayed.



5. Click the **Application Navigation** icon to open the left-navigation pane and click **Configuration** to open the **Database** tab of the **Configuration** page.



6. Click the plus (+) sign in the **Credentials** section to begin adding database user credentials.



7. You need to add connections for container database (CDB) and pluggable databases (PDBs). Each CDB is used to capture (Extract) from the source database and PDB for delivery (Replicat).

   Use the EZconnect syntax to configure the database connection. You need the username, password, hostname, port number, and service name connection information to use the EZConnect syntax.

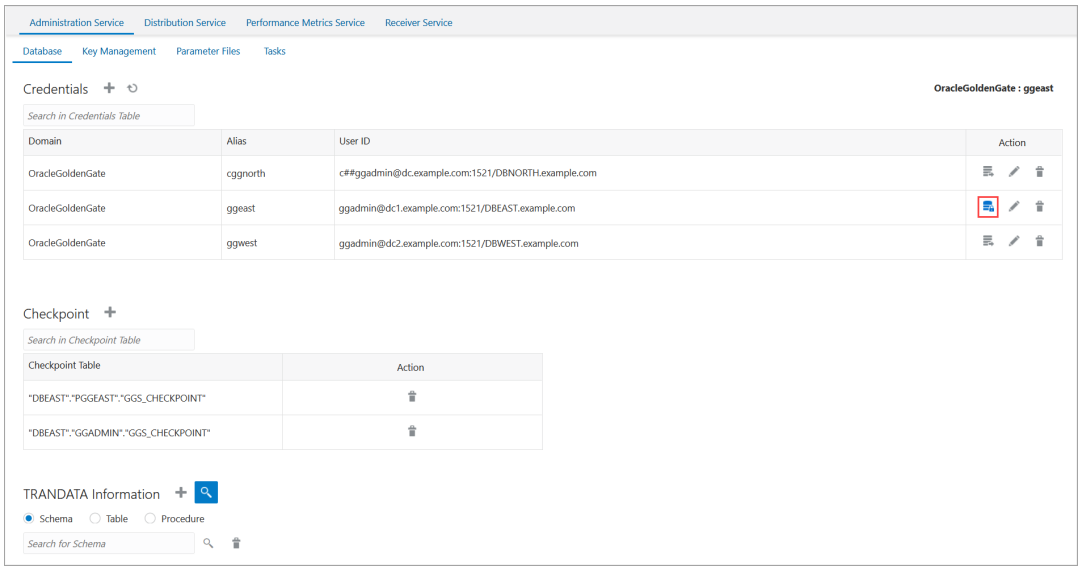
   Here's the syntax that you need to specify in the User ID field:

   ```
   username@hostname:port/service_name
   ```

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

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

8. Click on the blue icon in the **Actions** column to connect to the database. The icon turns blue when the connection is successful.



After connecting to the database, the sections to add checkpoint table, TRANDATA, and heartbeat table are displayed.

Step 2: Add Heartbeat, and Checkpoint Tables

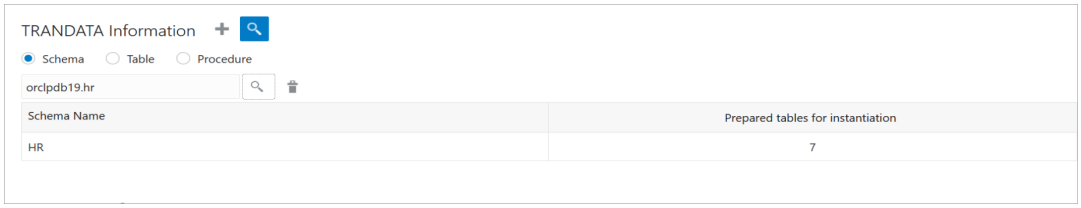
Add the heartbeat tables for the PDBs to monitor any possible lags.

Add a checkpoint table for the target database to ensure that if there is a failure, then the Extract and Replicat processes can restart from the point of failure.

> **Note:**
>
> You don't need to add TRANDATA as this is internally done with the PL/SQL call of `ADD_AUTO_CDR`. You might want to check that supplemental logging is enabled for the tables.

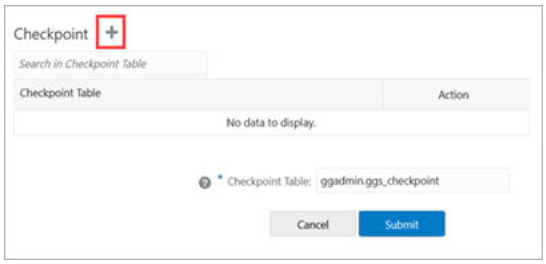
1. Use the **TRANDATA Information** section to check if supplemental logging has been enabled for the tables set up for capture.

   You can search for the schema for which you added the trandata, using the magnifier glass search icon. This will display the trandata information. The following image shows the trandata information for the `HR` schema in the pluggable database `DBEAST`.



See Configure Logging Properties to learn the steps for configuring the logging properties at the Schema, Table or Procedure level.

2. To set up the checkpoint table for Replicat, you need to connect to the target database credentials (**ggwest**) from the Credentials section.
3. Click the plus sign (+) to add the checkpoint table for the PDBs.



Click **Submit**. The checkpoint table is added.

Also see the Before Adding Extract and Replicat Processes section, for details on creating heartbeat tables.

4. Add another checkpoint table for the second Replicat, `reps`, by repeating the steps 3 and 4.
5. Add the heartbeat tables for both source and target endpoints by connecting to **ggeast** and **ggwest** database credential aliases.

   For bidirectional, active/active replication, the heartbeat table should be in the same schema for the outgoing Extracts and incoming Replicats at each site. For example, see the following use case:

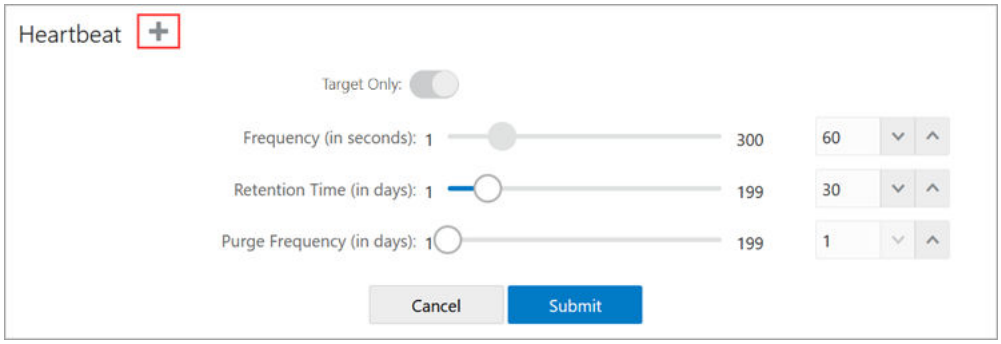
   Site A Site B

   `EAB ----------> RAB`

   `RBA ----------> EBA`

   In this example, `EAB` and `RBA` heartbeat tables must use the same schema. However, `EAB` and `RAB` can use different schemas.

   Add the heartbeat table by clicking the plus sign.

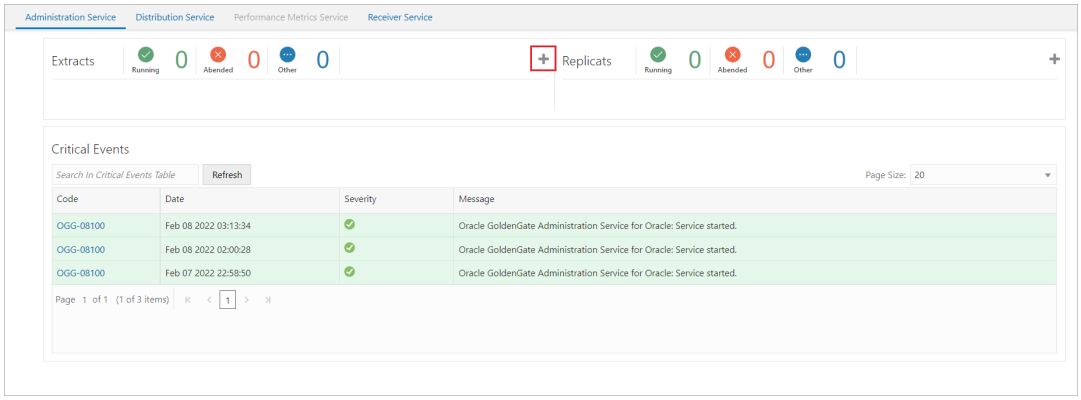


6. Click **Submit** after adjusting the heartbeat options.

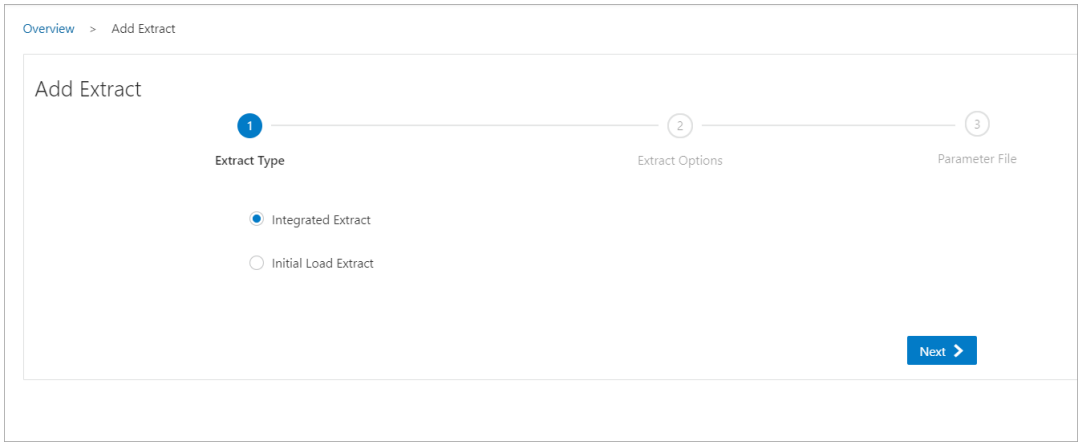
Step 3: Add Extracts

In this section, you will add two extracts, **`exte`** and **`extw`**. The Extract process captures data from the source database and writes it to a trail file. The trail file for **`exte`** is **`ea`** and for **`extw`** is **`ew`**.

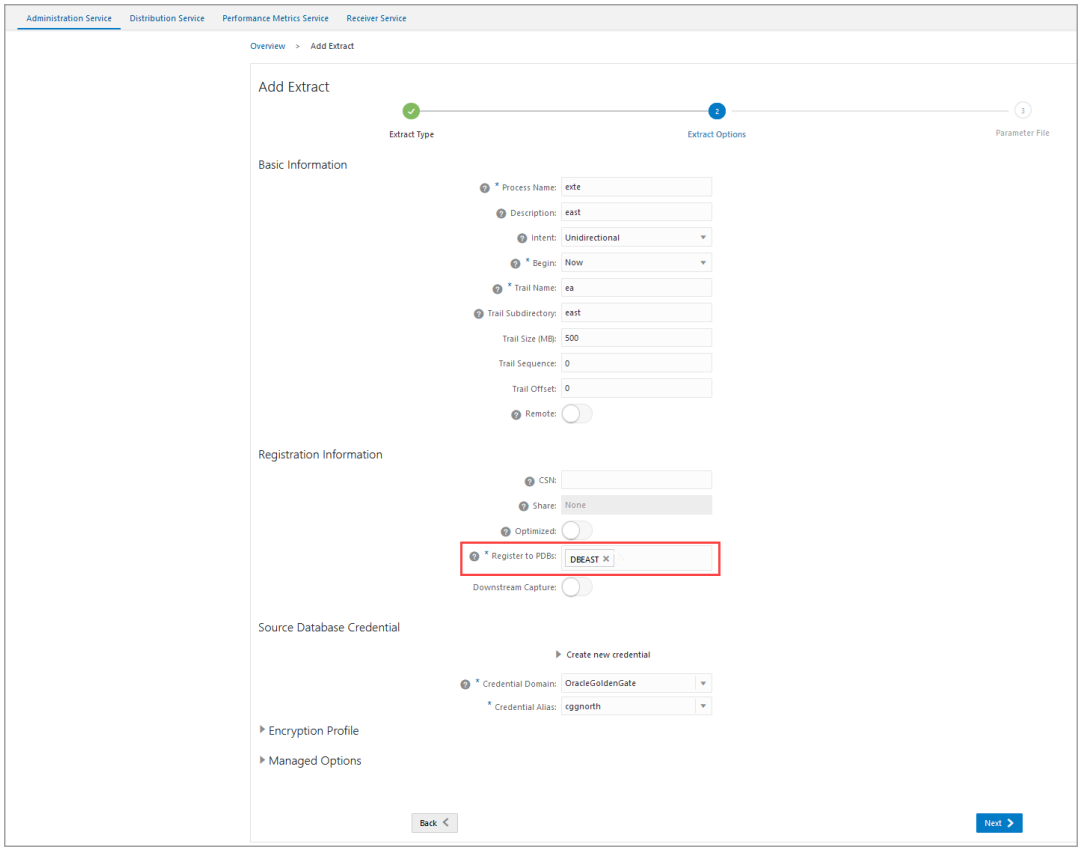
1. Click the **Overview** option from the left-navigation pane of the Administration Service and click the plus sign (+) from the Extract section.



2. From Add Extract wizard, select **Integrated Extract**.



3. Click **Next** and specify the Extract options in the Extract Options screen. See the detailed steps to add an Extract from the Add a Primary Extract section.



If you are creating the Extract for a pluggable database, then you'll see option **Register to PDBs** as soon as you enter the credentials domain and alias. Select the PDB in the container database that you want to use for replication.

4. After you enter the options for the Extract (`exte`), click **Next**. The next screen displays the Extract parameter file to help you review the Extract settings.

   Here's the Extract parameter file for the Extract exte:

```
EXTRACT exte
USERIDALIAS cggnorth DOMAIN OracleGoldenGate
EXTTRAIL east/ea
SOURCECATALOG DBEAST
TRANLOGOPTIONS EXCLUDETAG 00
DDL INCLUDE MAPPED OBJNAME hr.*
DDLOPTIONS REPORT
TABLE DBEAST.hr.*;
```

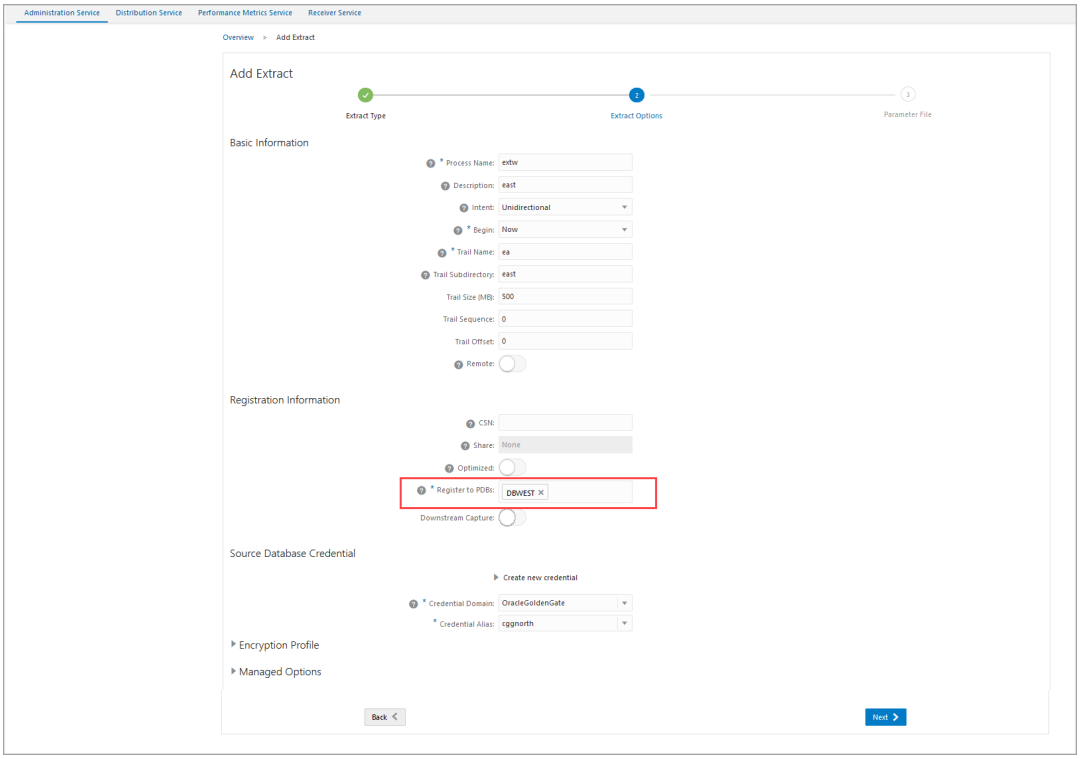
   Review these settings and update the Extract configuration as needed.

   For multitenant databases, you need to add entries for Extract to capture from multiple pluggable databases to a single trail. In the parameter file, source objects must be specified in `TABLE` statements with the fully qualified three-part names in the format of *`container.schema.object`* or using the `SOURCECATALOG` parameter with two-part names *`schema.object`*.

Click **Create and Run** to start your Extract.

To create the Extract **extw**:

1. Navigate back to the Overview page using the Application Navigation pane.
2. From Add Extract wizard, select Integrated Extract.
3. Click **Next** and specify the Extract options in the Extract Options screen.



4. Select the PDB as `DBWEST` in the container database that you want to use for replication.
5. After you enter the options for the Extract, click Next. The next screen displays the Extract parameter file to help you review the Extract settings.
6. Enter the options for Extract parameter:

```
EXTRACT extw
USERIDALIAS cggnorth DOMAIN OracleGoldenGate
EXTTRAIL west/ew
SOURCECATALOG DBWEST
TRANLOGOPTIONS EXCLUDETAG 00
DDL INCLUDE MAPPED OBJNAME hr.*
DDLOPTIONS REPORT
TABLE DBWEST.hr.*;
```

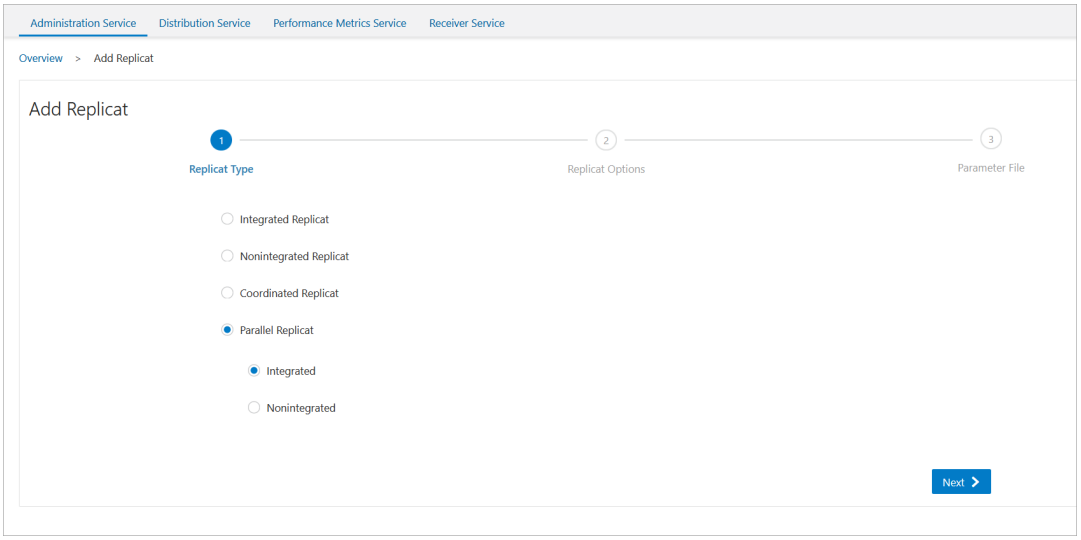
   Review these settings and update the Extract configuration as needed.

7. Click **Create and Run** to start your Extract.

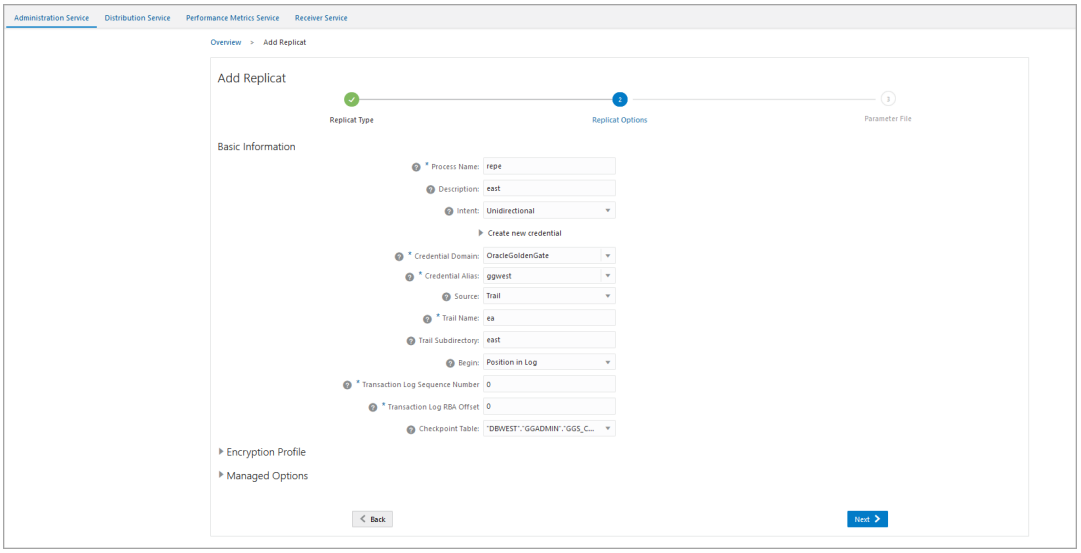
Step 4: Add a Replicat

In this section, you will add Replicats **repe** and **repw**. The Replicat process delivers the change data from the trail file (**ea**) created by the Extract, to the target database. Replicat reads the trail file on the target database, reconstructs the DML or DDL operations, and applies them to the target database.

1. Before you Add a Replicat, make sure that you added your checkpoint table for the target database (`DBWEST`) by connecting to the **ggwest** database credentials.
2. Select a Replicat type to deliver data to the target database. Follow the wizard to complete adding a Replicat.



3. Select the **Parallel Integrated Replicat** option in the Replicat Options screen.



4. Click **Next** to view the **Replicat Parameter File** screen. All the parameters that you have specified are available for review here.
   For multitenant container databases, Replicat can only apply to one pluggable database. To specify the correct one, use a SQL*Net connect string for the database user that you specify with the `USERID` or `USERIDALIAS` parameter. For example: `ggadmin@DBWEST`.

   In the parameter file, specify only the *`schema.object`* in the `TARGET` portion of the `MAP` statements. In the `MAP` portion, identify source objects captured from more than one pluggable database with their three-part names or use the `SOURCECATALOG` parameter with two-part names.

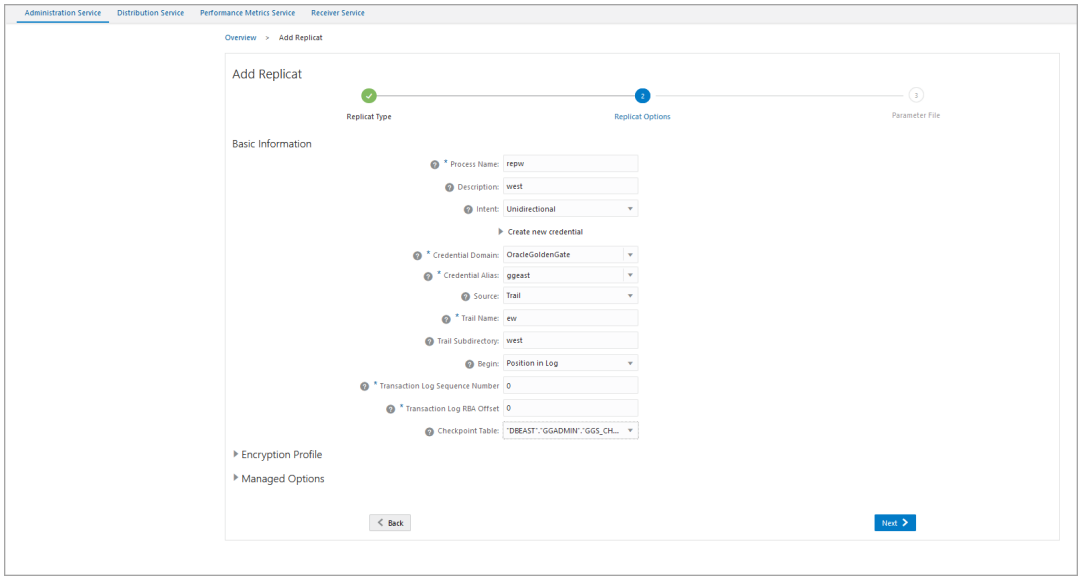
In case of integrated parallel Replicat, `MAPINVISIBLECOLUMNS` parameter is set by default. You don't need to set it in the Replicat parameter file explicitely.

Here's a sample of the Replicat Parameter File:

```
REPLICAT repe
USERIDALIAS ggwest DOMAIN OracleGoldenGate
DDLOPTIONS REPORT
SOURCECATALOG DBEAST
MAP hr.*, TARGET hr.*;
```

To create the second Replicat **repw**, follow these steps:

1. Repeat steps 1 and 2 from the steps to add the first Replicat (**repe**).
2. In the Replicat options screen, enter the following details:



Apart from entering the other options, make sure you enter the following details:

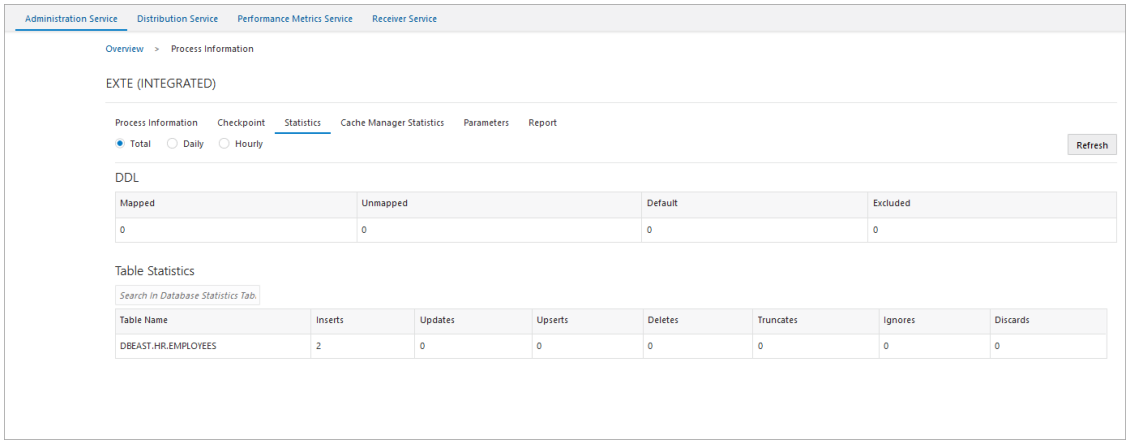
a. Specify the trail name as **ew** and the trail file subdirectory as west.
b. Select the checkpoint table as `DBWEST.ggs_checkpoint`.
c. Click Next.
d. Change or modify the Replicat parameter file, as follows:

```
REPLICAT repw
USERIDALIAS ggeast DOMAIN OracleGoldenGate
SOURCECATALOG DBWEST
DDL INCLUDE ALL
DDLOPTIONS REPORT
MAPEXCLUDE ggadmin.ggs_checkpoint*
MAPINVISIBLECOLUMNS
MAP hr.*, TARGET hr.*;
```

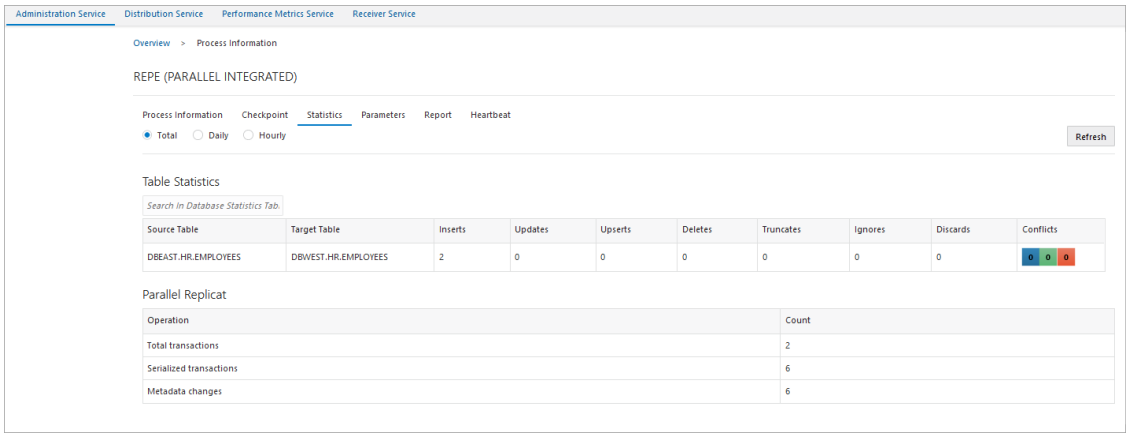
After the Replicat starts successfully, you can see the Extract and Replicat processes in running state on the **Administration Service** Overview page.

Test and Monitor Transactions

The following screen shows that records were captured by the **`exte`** Extract from the `hr.employees` table on `DBEAST`.

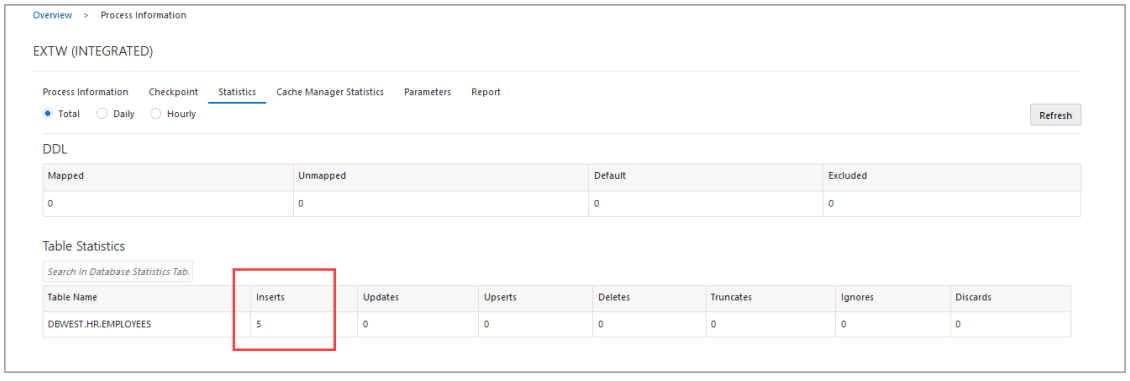


Check that the same is updated on the Replicat (**`repe`**) as well:



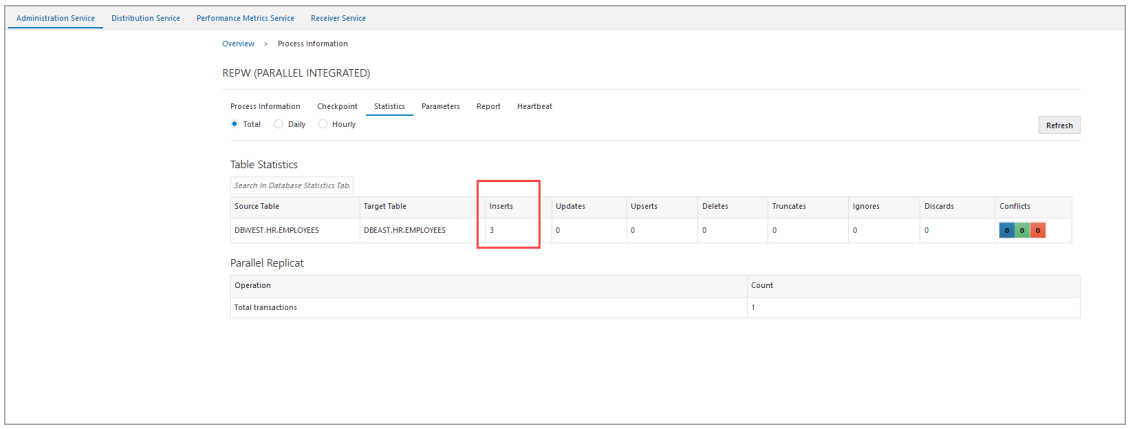
The 2 records in the `hr.employees` table are replicated to the endpoint (`DBWEST`).

Let's see the Extract (**`extw`**) on `DBWEST`.



Notice that the **value of inserted records is 5**. Out of these 5 records, **2 were replicated by `repe`** into `hr.employees` on **`DBWEST`**. **3 new records** were then inserted into `hr.employees` on **`DBWEST`**.

When these 3 records are inserted in the `hr.employees` table in the PDB `DBWEST`, then only the updated records should be replicated in `DBEAST`. The following screen shows that only the updated records are added to `DBEAST`.



As shown in this figure, there are 3 INSERTS, indicating that there was no duplication of records.

This is one way of implementing an active-active bidirectional replication in Oracle GoldenGate MA.

Test Automatic Conflict Detection and Resolution

In this section, the latest timestamp of a record is checked to check if ACDR is able to resolve the conflict in records. To check automatic resolution of conflicts, let's create the following records.

Transaction in `DBEAST`:

In the following example, UPDATE transactions have been run simulateneously on `DBEAST` and `DBWEST` and with ACDR, the conflict is detected and resolved.

Here's the query to update a records in hr.employees on `DBEAST`:

```
UPDATE hr.employees set LAST_NAME='Simmonds', EMAIL='HSIMMONDS' where
EMPLOYEE_ID=204;


UPDATE hr.employees set SALARY='15000' where EMPLOYEE_ID=203;
```

Simulteneously, another query is run on `DBWEST` for the same rows, as shown in the following example:

```
UPDATE hr.employees set LAST_NAME='Symmonds', EMAIL='HSYMMONDS' where
EMPLOYEE_ID=204;


UPDATE hr.employees set SALARY='25000' where EMPLOYEE_ID=203;
```

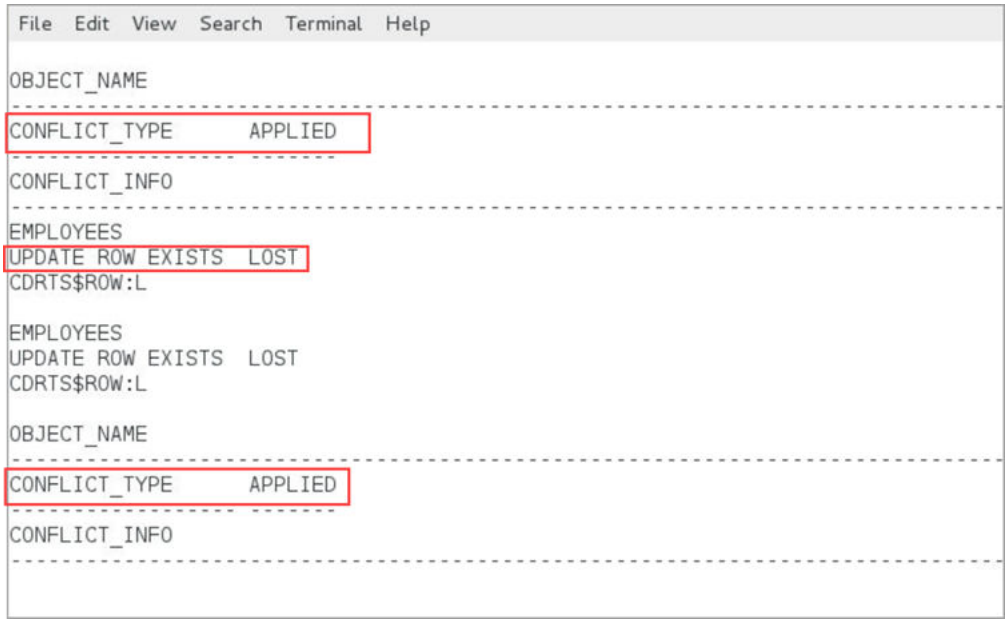
To check which of these entries was the winner or the entry that was finally applied, and to know the criteria used to apply that entry, use the following options:

Use DBA_APPLY_ERROR_MESSAGES view

On DBEAST, run the following query:

```
select OBJECT_NAME, CONFLICT_TYPE,APPLIED_STATE,CONFLICT_INFO from
DBA_APPLY_ERROR_MESSAGES;
```

The output for this query displays the following:



Run the same query on DBWEST also.

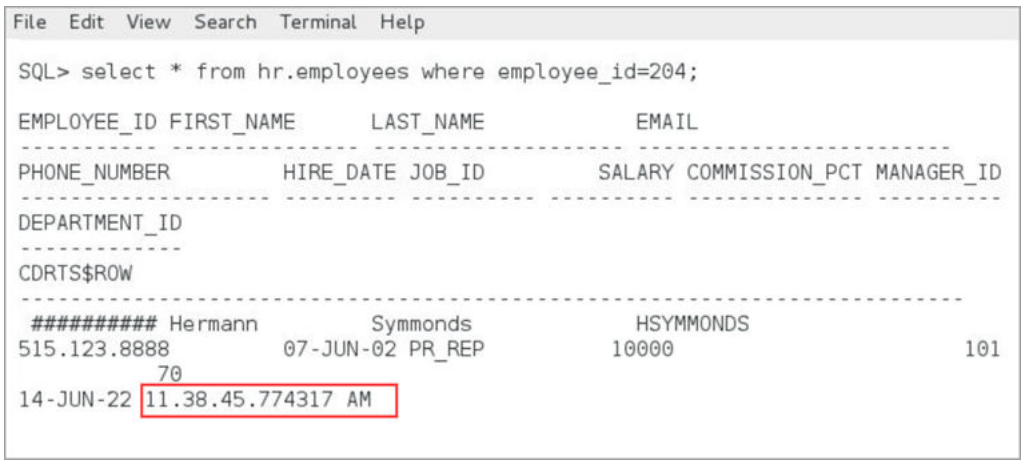
Make the CDRTS$ROW visible by running the following command:

```
ALTER TABLE hr.employees modify CDRTS$ROW visible;
```

To check the record that was eventually applied to the rows, run the SELECT query on the table hr.employees. You can run this query on either DBEAST or DBWEST:

```
SELECT * from hr.employees WHERE employee_id=204
```

The output shows as follows:



You can note the timestamp for this transaction: 11.38.45.774317 AM.

Now, let's check the timestamp on **DBWEST**:

```
SQL> select * from hr.employees where employee_id=204;

EMPLOYEE_ID FIRST_NAME      LAST_NAME            EMAIL
----------- --------------- -------------------- -------------------------
PHONE_NUMBER         HIRE_DATE JOB_ID         SALARY COMMISSION_PCT MANAGER_ID
-------------------- --------- ---------- ---------- -------------- ----------
DEPARTMENT_ID
-------------
CDRTS$ROW
--------------------------------------------------------------------------------
 ########## Hermann         Symmonds             HSYMMONDS
515.123.8888         07-JUN-02 PR_REP          10000                     101
           70
14-JUN-22 11.38.45.774317 AM
```

As the conflict is resolved, the timestamp shows the same data on both PDBs.

Run the following command to check if the conflicts were detected and resolved on the Oracle GoldenGate side. Here's the command to check this:

```
STATS REPLICAT repe, REPORTCDR
```

The output for this command displays the following:

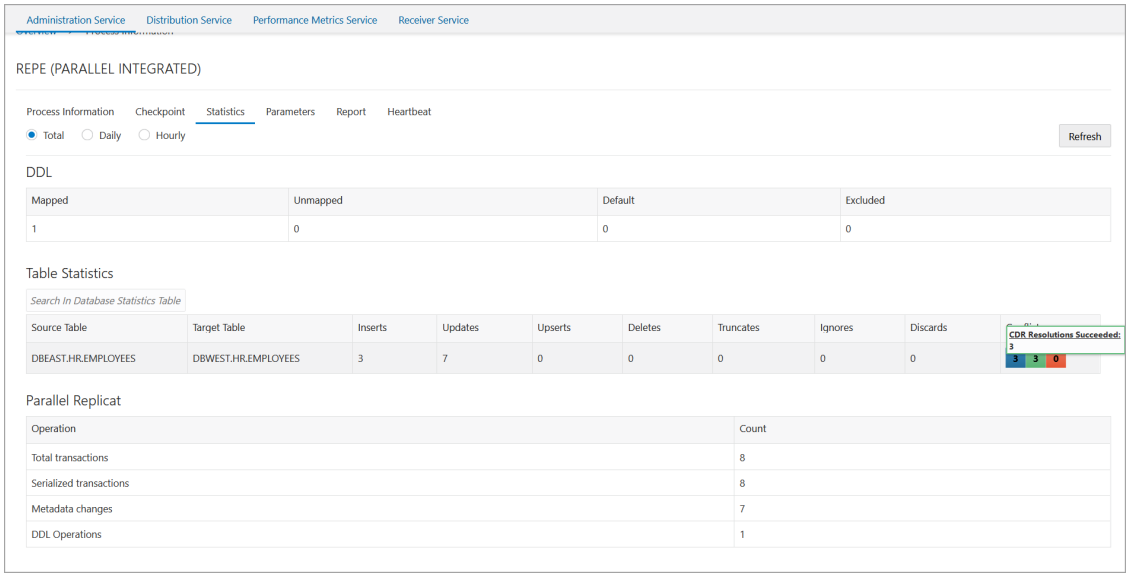
```
Replicating from DBEAST.HR.EMPLOYEES to DBWEST.HR.EMPLOYEES:

*** Total statistics since 2022-06-07 06:13:19 ***
    Total inserts                                3.00
    Total updates                                7.00
    Total deletes                                0.00
    Total upserts                                0.00
    Total discards                               0.00
    Total operations                            10.00
    Total CDR conflicts                          3.00
    CDR resolutions succeeded                    3.00
    CDR UPDATEROWEXISTS conflicts                3.00

*** Daily statistics since 2022-06-14 00:00:00 ***
    Total inserts                                0.00
    Total updates                                2.00
    Total deletes                                0.00
    Total upserts                                0.00
    Total discards                               0.00
    Total operations                             2.00
    Total CDR conflicts                          2.00
    CDR resolutions succeeded                    2.00
    CDR UPDATEROWEXISTS conflicts                2.00
```

As shown in this statistical report, there were 3 conflicts and 2 of them were resolved. The `UPDATEROWEXISTS` conflict type is used for resolution.

You can also see this report from the web interface:



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 exisiting 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 `checkpointtable` 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 exisitng 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/ea000000009
                     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/ea000000009                        2022-06-16 04:21  2022-06-14
04:38
  east/ea000000000                        * Initialized *   2022-06-16
04:21
  east/ea000000009                        2022-06-14 04:38  2022-06-14
04:38
  east/ea000000009                        2022-06-14 04:38  2022-06-14
04:38
  east/ea000000009                        2022-06-16 03:55  2022-06-14
04:38
  east/ea000000000                        * Initialized *   2022-06-16
03:55
  east/ea000000000                        * Initialized *   First
Record
  east/ea000000000                        * Initialized *   First
Record
  east/ea000000000                        * Initialized *   First
Record
  east/ea000000000                        * 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.

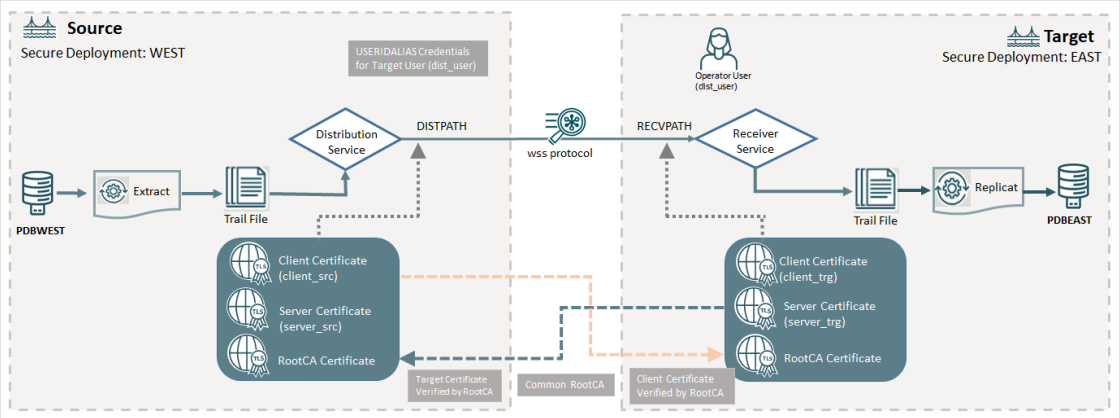
Connecting Two Deployments Using a Common RootCA Certificate

There are multiple approaches which you implement for applying certificates when working across different source and target deployments. This quickstart demonstrates how to set up and apply certificates when using a common RootCA certificate on the source and target deployment.

Here is an outline of the process that's followed for configuring certificates when using a common rootCA certificate:

1. Create and add client, server, and rootCA certificates to the source and target deployment. Ensure that the same rootCA certificate is available on both source and target deployments.
2. Create a user on the target deployment with Operator role and user Type set to Password. Add credentials for this user to the source deployment to allow connection between source and target deployment where the target authentication method is the password of the created user.

As shown in the following diagram, secure source and target deployments have client and server certificates and a common rootCA certificate. Authorizing an Oracle GoldenGate deployment user with role as Operator and user type as Password to set up credential alias on source deployment to connect to target.



Create Client, Server, and Trusted Chain Certificates

This part of the quickstart is a **prerequisite** to Connect the Two Deployments through the Distribution Path.

In a production environment, certificates are provided by a digital certificate authority such as DigiCert. However, in a testing environment, you can generate certificates using the instructions in this quickstart.

> **Note:**
>
> Make sure that the latest JDK is installed and the `JAVA_HOME` environment variable is set up. This is required to run the orapki utility.

Follow the steps in this topic, to create server, client, and trusted certificate chains on the source and target deployments.

Source Deployment:

1. Create a rootCA certificate.
   a. Use the configuration file similar to the following, for rootCA.cfg:

```
[ 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 = "gg-Root"
[ v3_req ]
basicConstraints=CA:TRUE
[ v3_ca ]
basicConstraints=CA:TRUE
[ usr_cert ]
```

```
basicConstraints=CA:TRUE
[ my_extensions ]
```

b. Use the following command to create the rootCA certificate:

```
openssl req -x509 -newkey rsa:4096 -keyout rootCA.key -out rootCA.cert -
days 73000 -nodes -config
rootCA.cfg
```

2. Create a Server Certificate.

a. Make a directory `server` and navigate to this directory.

```
mkdir server
cd server
```

b. View the server configuration file, `server.cfg`

```
[ req ]
default_bits = 4096
default_md = sha512
prompt = no
encrypt_key = no
distinguished_name = req_distinguished_name
[ req_distinguished_name ]
commonName = "west.oracle.com"
[ my_extensions ]
```

c. Create the server certificate.

```
openssl req -new -newkey rsa:2048 -nodes -keyout servery -out servery -
config server.cfg

openssl x509 -req -days 73000 -in server.csr -CA ../rootCA.cert -
CAkey ../rootCA.key -CAcreateserial -out
server.cert
```

3. Create an empty auto-login Oracle wallet for Server.

```
orapki wallet create -wallet ../server -auto_login
```

When prompted, enter the password for logging in to the wallet.

4. Create a `pkcs#12` file using the server certificate and trusted certificate (rootCA) chain details.

```
openssl pkcs12 -export -out server.p12 -inkey server.key -in server.cert -
chain -CAfile ../rootCA.cert
```

Provide the password, if prompted.

5. Import the `pkcs#12` file into the auto-login wallet.

```
orapki wallet import_pkcs12 -wallet ../server -pkcs12file ./server.p12
```

Enter the password when prompted.

6. Review the content of the wallet.

```
orapki wallet display -wallet ../server/ -complete
```

> **Note:**
>
> Make sure that you have the latest JDK installed on the system.

7. Create the client configuration file and client wallet with the client certificate.
   a. Make directory client.

```
cd ..
mkdir client
cd client
cat >client.cfg <<EOF
```

Client configuration file (client.cfg):

```
[ req ]
default_bits = 4096
default_md = sha512
prompt = no
encrypt_key = no
distinguished_name = req_distinguished_name
[ req_distinguished_name ]
commonName = "client_src"
[ my_extensions ]
EOF
```

   b. Create the client certificate, using the steps shown in the following code snippet:

```
openssl req -new -newkey rsa:2048 -nodes -keyout client.key -out
client.csr -config client.cfg
openssl x509 -req -days 73000 -in client.csr -CA ../rootCA.cert -
CAkey ../rootCA.key -CAcreateserial -out
client.cert
```

   c. Create an empty auto-login wallet for the client.

```
orapki wallet create -wallet ../client -auto_login
```

   d. Create a `pkcs#12` file using the client certificate and trusted certificate (rootCA) chain details.

```
openssl pkcs12 -export -out client.p12 -inkey ./client.key -in ./
client.cert -chain -CAfile ../rootCA.cert
```

   e. Import the `pkcs#12` file into the auto-login wallet.

```
orapki wallet import_pkcs12 -wallet ../client -pkcs12file ./client.p12
```

f. Review the content of the client wallet.

```
orapki wallet display -wallet ../client/ -complete
```

Target Deployment

Repeat the same steps to create the target server and client certificate, as were used for generating the source server and client certificate.

1. Copy the rootCA certificate created previously, to the target deployment.
2. Generate the target server certificate, as shown in the following code snippet:

```
## a. Make target directory
cd ..
mkdir target
cd target

## b. Create target configuration file
cat > target.cfg << EOF

## c. Target Configuration File (target.cfg)
[ req ]
default_bits = 4096
default_md = sha512
prompt = no
encrypt_key = no
distinguished_name = req_distinguished_name
[ req_distinguished_name ]
commonName = "east.oracle.com"
[ my_extensions ]
EOF

## d. Create the certificate for the target server
openssl req -new -newkey rsa:2048 -nodes -keyout target.key -out
target.csr -config target.cfg
openssl x509 -req -days 73000 -in target.csr -CA ../rootCA.cert -CAkey ../
rootCA.key -CAcreateserial -out
target.cert

## e. Create an empty auto-login wallet for the target certificate
orapki wallet create -wallet ../target -auto_login

## f. Create a pkcs#12 file using the target certificate and trusted
certificate chain details
openssl pkcs12 -export -out target.p12 -inkey ./target.key -in ./
target.cert -chain -CAfile ../rootCA.cert

## g. Import the pkcs#12 file into the auto-login wallet
orapki wallet import_pkcs12 -wallet ../target -pkcs12file ./target.p12

## h. Review the content of the wallet
orapki wallet display -wallet ../target/ -complete
```

3. Generate the target client certificate:

```
## make directory
cd ..
mkdir client
cd client
cat >client.cfg <<EOF


## client configuration file
[ req ]
default_bits = 4096
default_md = sha512
prompt = no
encrypt_key = no
distinguished_name = req_distinguished_name
[ req_distinguished_name ]
commonName = "client_trg"
[ my_extensions ]
EOF


##command to generate the client certificate
openssl req -new -newkey rsa:2048 -nodes -keyout client.key -out
client.csr -config client.cfg
openssl x509 -req -days 73000 -in client.csr -CA ../rootCA.cert -CAkey ../
rootCA.key -CAcreateserial -out
client.cert


## Create an empty auto-login wallet for the target certificate
orapki wallet create -wallet ../client -auto_login

## Create a pkcs#12 file using the target certificate and trusted
certificate chain details.
openssl pkcs12 -export -out client.p12 -inkey ./client.key -in ./
client.cert -chain -CAfile ../rootCA.cert

## Import the pkcs#12 file into the auto-login wallet
orapki wallet import_pkcs12 -wallet ../client -pkcs12file ./client.p12

## Review the content of the wallet
orapki wallet display -wallet ../client/ -complete
```

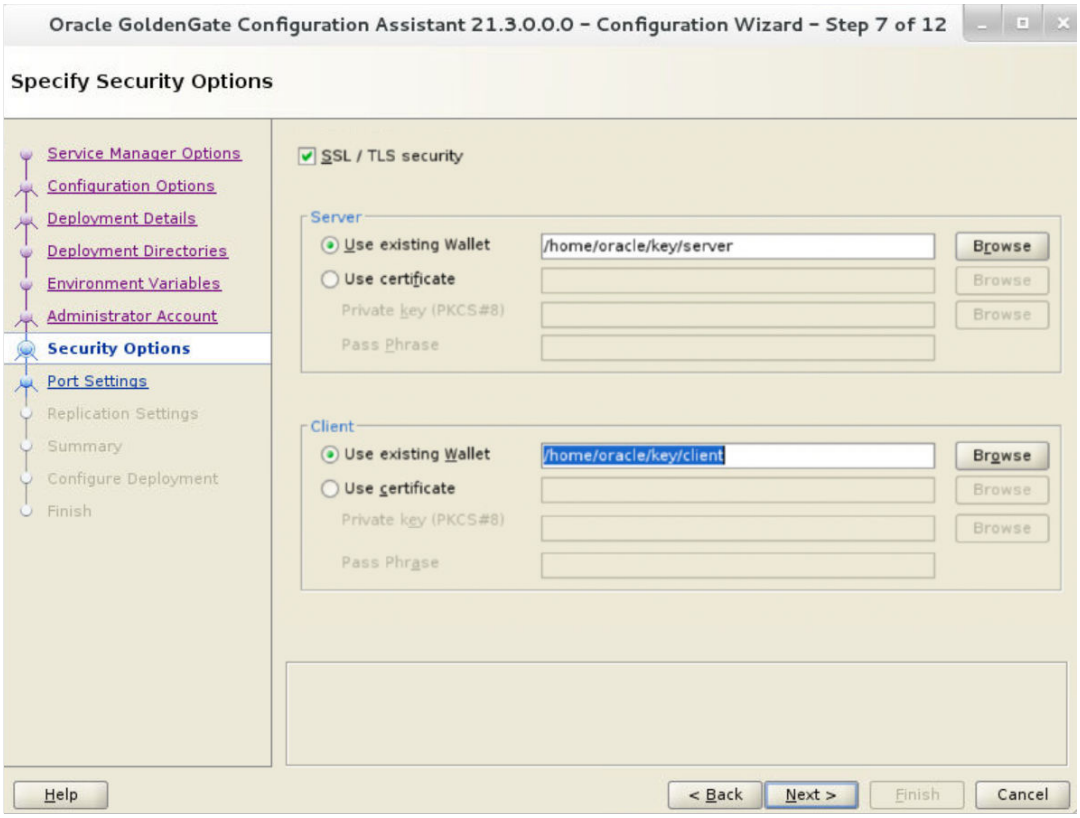
Apply Certificates to Source and Target Deployments

Oracle GoldenGate provides two ways for applying certificates for deployments:

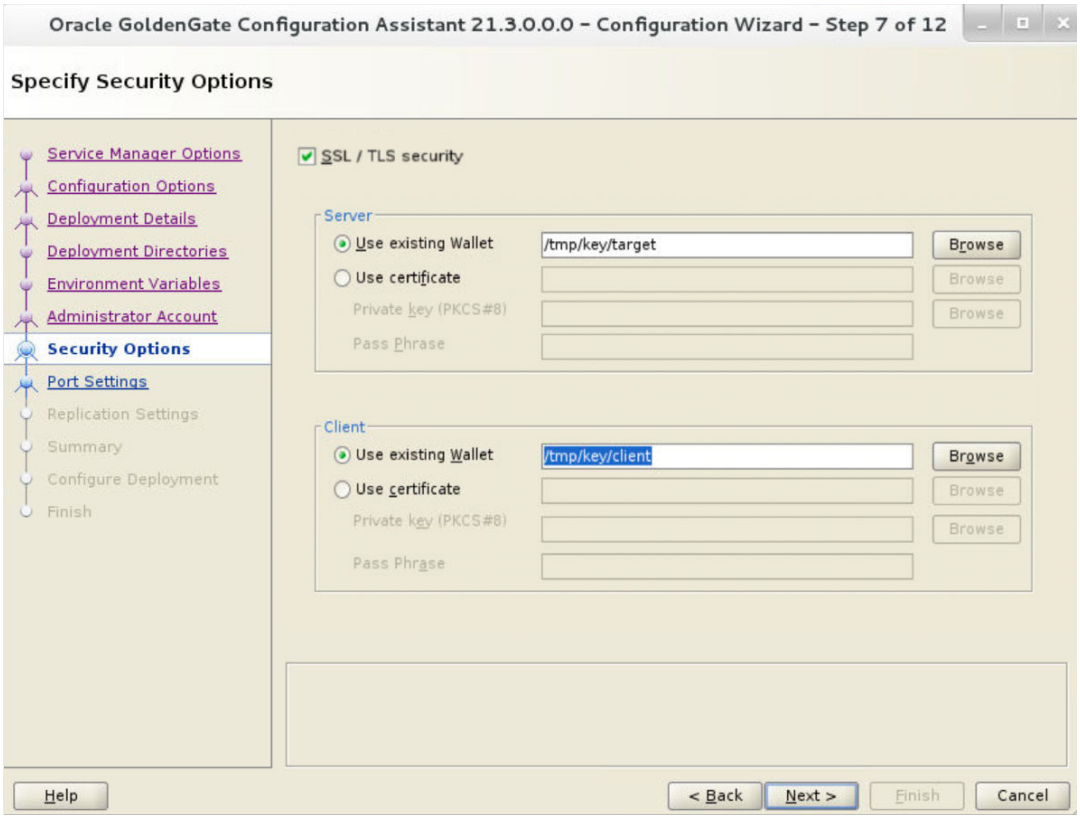
- Applying Certificates When Creating a Secure Deployment Using OGGCA: Use this option if you have existing wallets and certificates on source and target deployments.
- Applying Certificates Using the Service Manager Certificate Management page: Use this option when there are existing deployments, where certificates need to be applied on the source and target deployments.

In this quickstart, the OGGCA method is used to apply certificates while setting up a secure deployment.

1. On the **source**, run the OGGCA wizard and provide the wallet details on the Specify Security Options screen.



On the **target** also, run the OGGCA utility and provide the wallet location of the target server and client wallets on the Specify Security Options screen.



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.

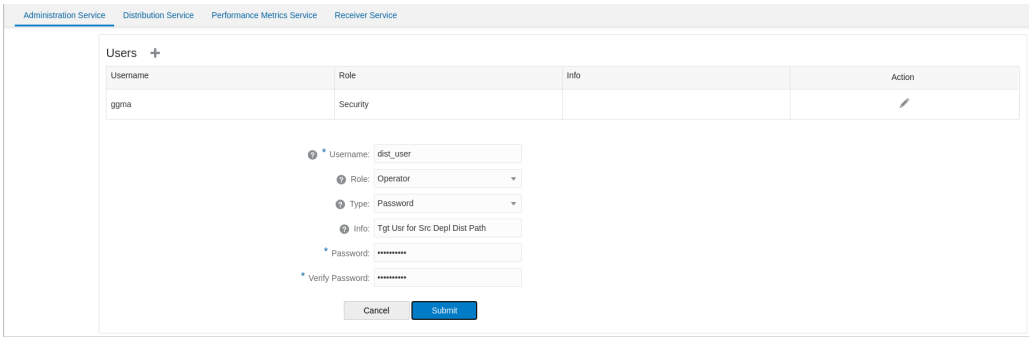
Connect the Two Deployments through the Distribution Path

This part of the quickstart describes the tasks that you need to perform if you need to use server, client, and trusted chain certificates when connecting two different deployments over a secure network.

After applying certificates, do the following:

1. Set up the distribution path on **source** deployment.
2. Create a user with the Operator role on **target** deployment, which would be used by the source when the DISTPATH establishes connection with the target deployment.
   a. From the Administration Service, navigate to the Administrator menu.
   b. Click the plus sign next to Users.

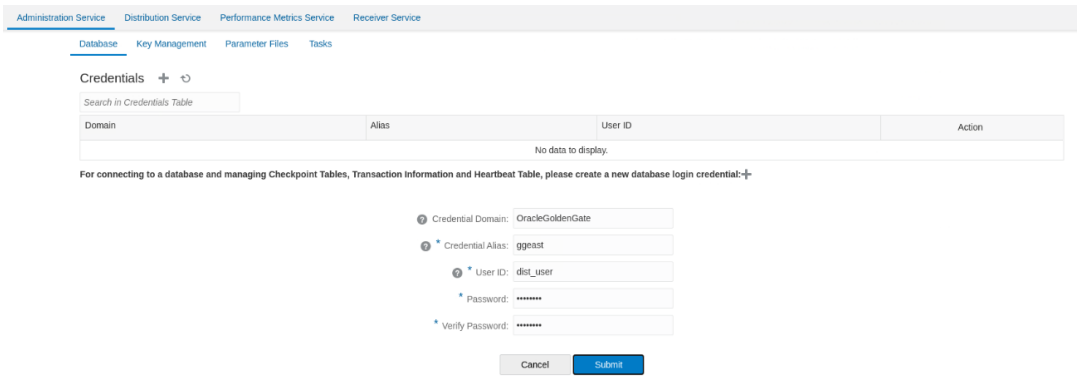
c. Specify the details for target deployment Operator user, as shown in the following image:



d. Specify a password and click **Submit**.

Configure the Distribution Path for Connecting Source to Target

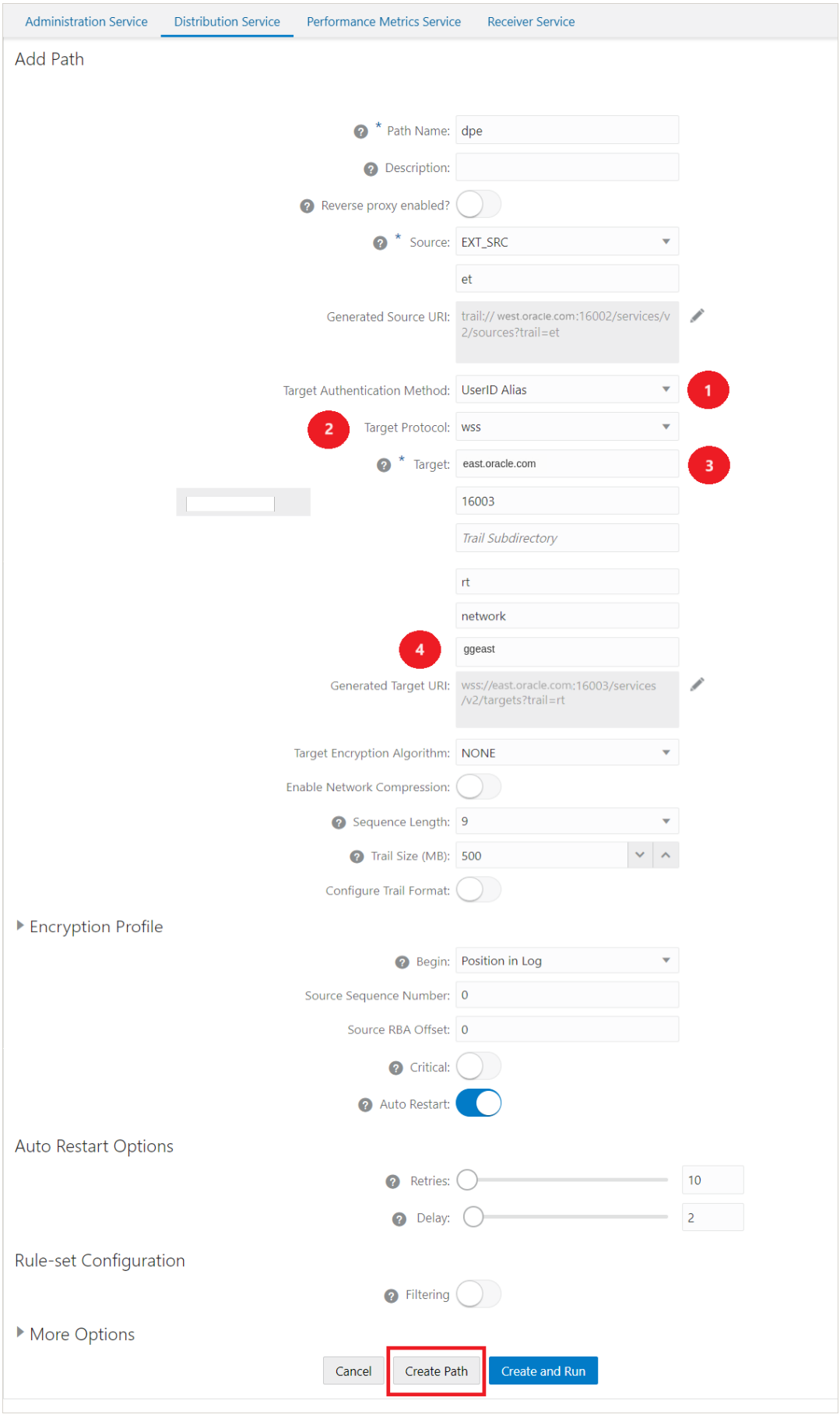
1. On the target side, add a user with the Operator role (**dist_user**) that is used by the source deployment when the DISTPATH connects to the target deployment.
2. On the source side, create a credential alias to store the Operator user credential that was created on the target side (dist_user).



3. Create a distribution path on the **source** side.

> **Note:**
>
> Make sure to use the following guidelines when creating the distribution path:
>
> - Select the **wss** protocol.
> - Use the **Target Authentication Method** as **UserID Alias** and specify the credential alias, ggeast, which was created for the Operator user.
> - Make sure that the target hostname matches the CN name or SAN name(s) in the target certificate. In this case, the value is **east.oracle.com**. If this value does not match the value specified in the target configuration file, then the connection will not work.

See the following image to see the options to set up the configuration:


Administration Service
Distribution Service
Performance Metrics Service
Receiver Service
Add Path
* Path Name: dpe
Description:
Reverse proxy enabled?
* Source: EXT_SRC
et
Generated Source URI: trail:// west.oracle.com:16002/services/v2/sources?trail=et
Target Authentication Method: UserID Alias
1
2
Target Protocol: wss
* Target: east.oracle.com
3
16003
Trail Subdirectory
rt
network
4
ggeast
Generated Target URI: wss://east.oracle.com:16003/services/v2/targets?trail=rt
Target Encryption Algorithm: NONE
Enable Network Compression:
Sequence Length: 9
Trail Size (MB): 500
Configure Trail Format:
Encryption Profile
Begin: Position in Log
Source Sequence Number: 0
Source RBA Offset: 0
Critical:
Auto Restart:
Auto Restart Options
Retries: 10
Delay: 2
Rule-set Configuration
Filtering
More Options
Cancel
Create Path
Create and Run

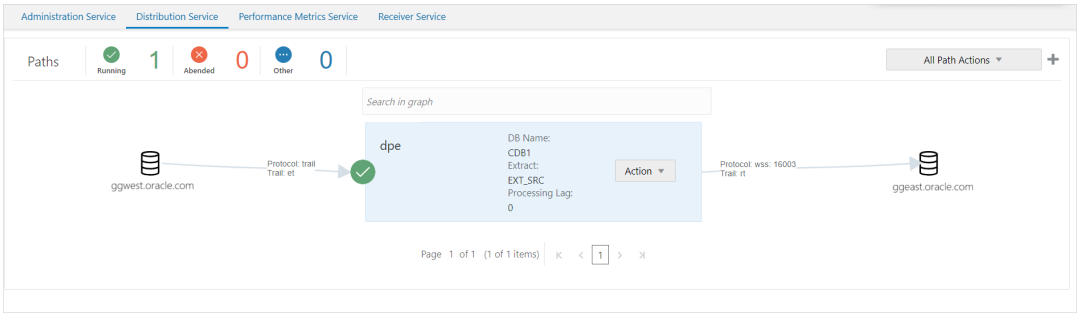
1: Target Authentication Method is selected as UserID Alias

2: Target protocol is selected as **wss**

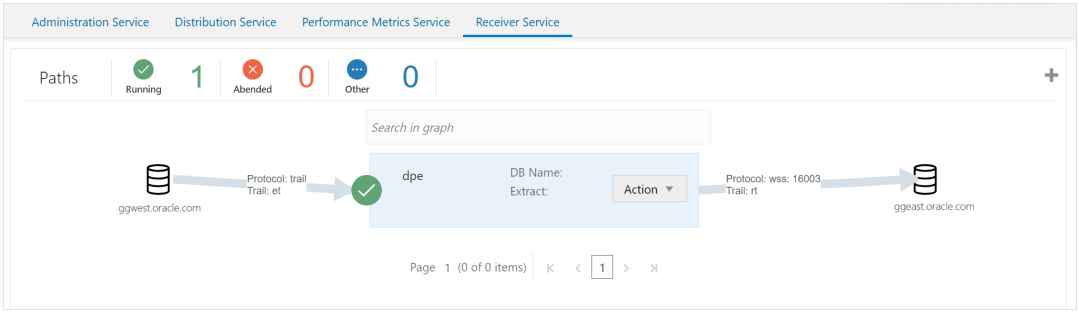
3: Target name is the same as the CN name specified in the target.cfg file

4: Credential alias for the Operator user created in the target deployment, is added as alias

4. Click **Create Path**. The path is created successfully and is visible on the Distribution Service home page.
5. To start the path, click **Action** and then click **Start**. This will start the distribution path as shown in the following image. You will be able to see the distribution path in running state.



6. Navigate to the Receiver Service and you will see the distribution path in running state.



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.

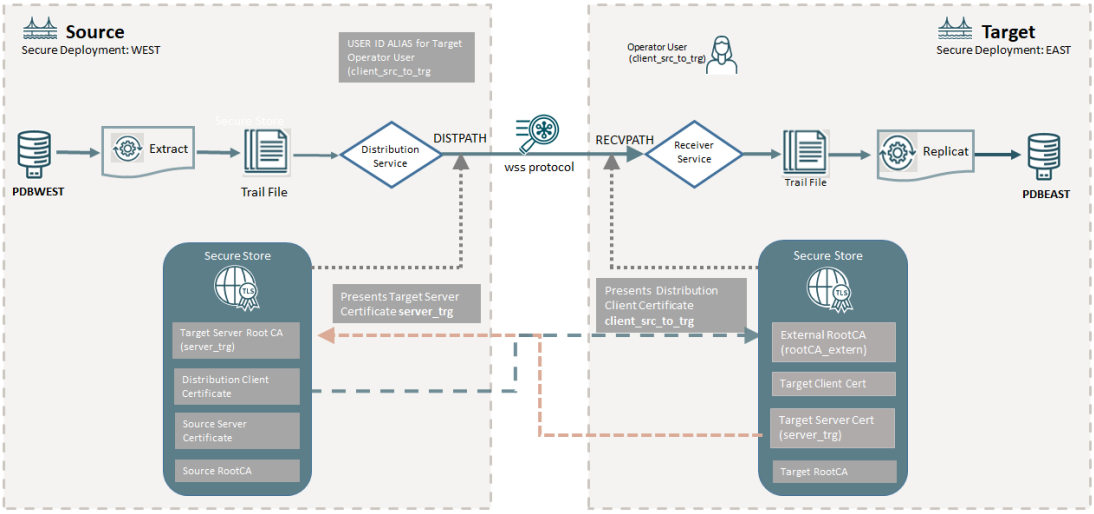
Environment

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_extern certificate:
   a. Use a configuration file similar to the following for rootCA_extern:

```
[ 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_extern"
[ 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_extern certificate:

```
# rootCA certificate
openssl req -x509 -newkey rsa:4096 -nodes       \
            -keyout rootCA_extern.key           \
            -out rootCA_extern.cert -days 73000 \
            -config rootCA_extern.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

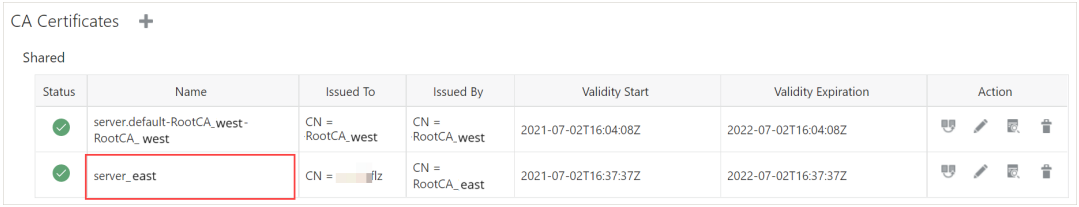
Oracle GoldenGate provides two ways for applying certificates for deployments:

- 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).

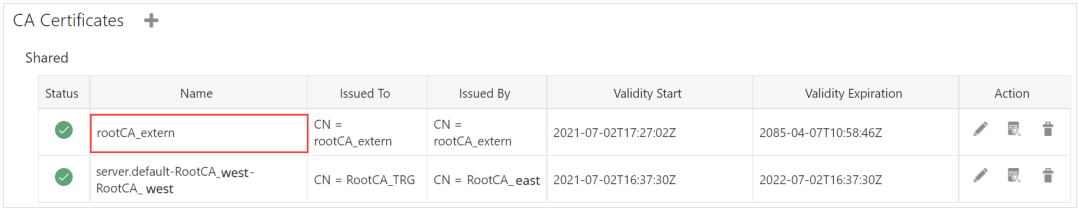
   The Distribution service on the source system must trust the target server certificate, which is authorized by the server_east target server CA certificate. This is added to the source secure store.



   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.

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.



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.

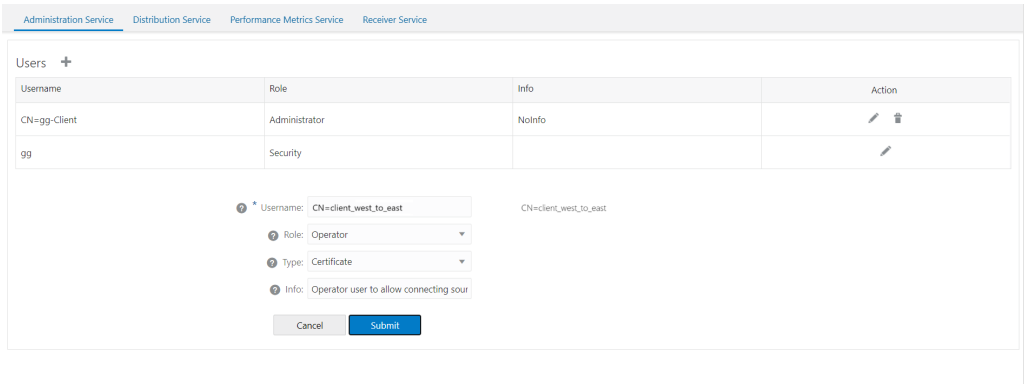
After the certificates are added on the source and target deployment, you can configure the distribution path on the source deployment to connect to the target.

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, navigate to the Administrator menu.
   b. Click the plus sign next to Users.
   c. Specify the details for target deployment Operator user, as shown in the following image:

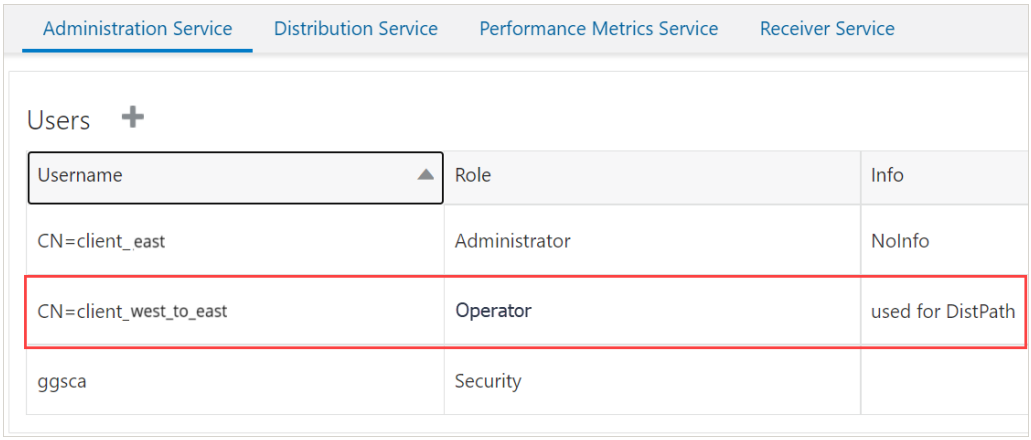


Notice that the username is set as the **CN** value mentioned in the distribution path client certificate.

> **Note:**
>
> In this quickstart, the **Certificate** type of user authentication is used. You can also use the **Password** type for authentication, as discussed in Set Up the Password Type User Authentication.

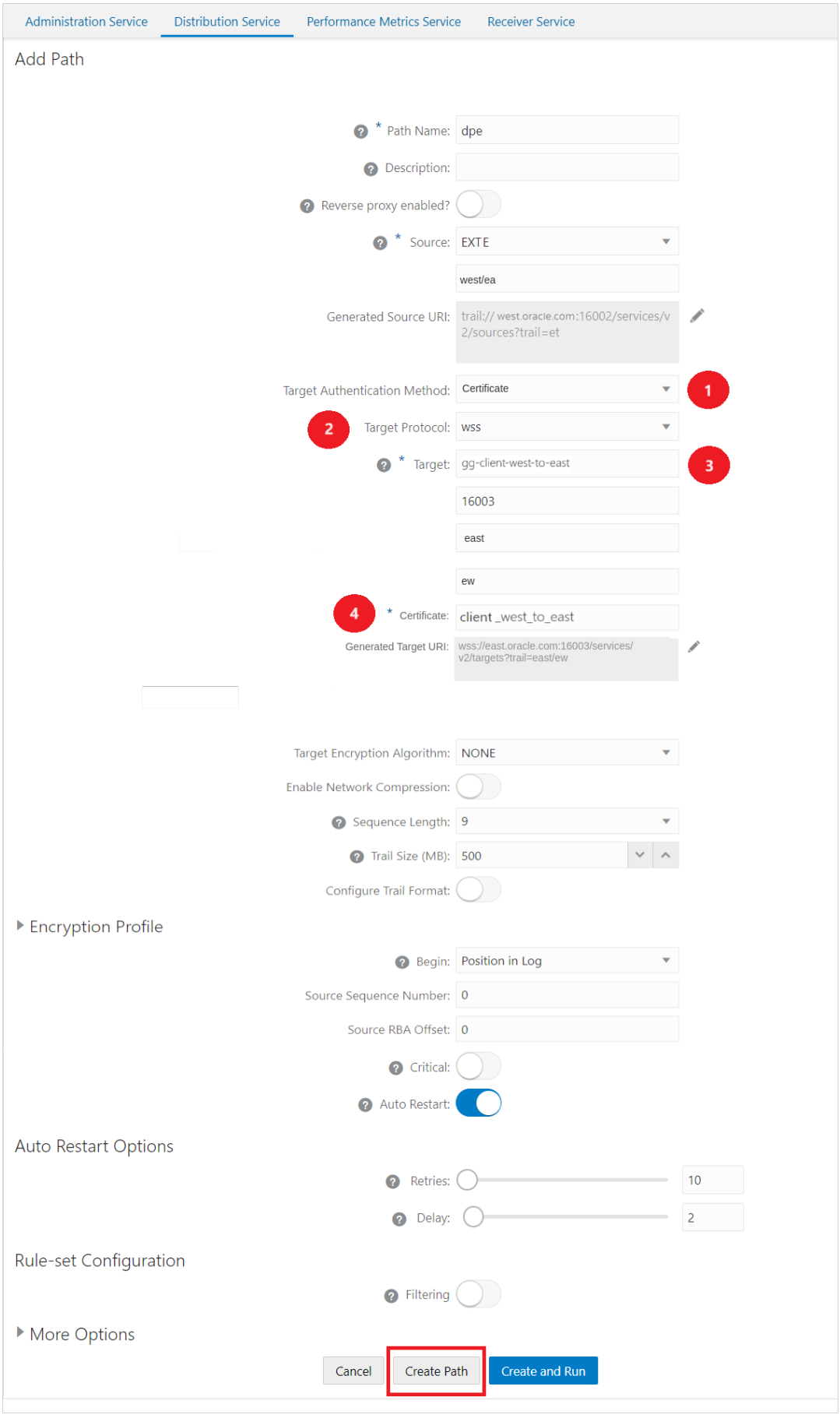
d. Specify a password and click **Submit**.

The user is created as shown in the following image:



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 diagram displays 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 image displays the distribution path options while adding a distribution path from the Distribution Service:


Administration Service
Distribution Service
Performance Metrics Service
Receiver Service
Add Path
Path Name: dpe
Description:
Reverse proxy enabled?
Source: EXTE
west/ea
Generated Source URI: trail:// west.oracle.com:16002/services/v2/sources?trail=et
Target Authentication Method: Certificate
1
2
Target Protocol: wss
Target: gg-client-west-to-east
3
16003
east
ew
4
Certificate: client _west_to_east
Generated Target URI: wss://east.oracle.com:16003/services/v2/targets?trail=east/ew
Target Encryption Algorithm: NONE
Enable Network Compression:
Sequence Length: 9
Trail Size (MB): 500
Configure Trail Format:
Encryption Profile
Begin: Position in Log
Source Sequence Number: 0
Source RBA Offset: 0
Critical:
Auto Restart:
Auto Restart Options
Retries: 10
Delay: 2
Rule-set Configuration
Filtering
More Options
Cancel
Create Path
Create and Run

In the given image:

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.

6
Extract

Learn about different types of Extract and how to add and manage Extracts.

About Extract

Extract is a process that is configured to run against the source database or configured to run on a downstream mining database (Oracle only) with 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. See Add Initial Load Extract Using the Admin Client and Replicat: Advance Tasks.
- **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 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.

> **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.

The Extract process can be configured 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, which are used for supplemental logging.

Add Extracts

Learn about adding different type of Extract(s), depending on the specific requirement, and database used with Oracle GoldenGate.

Add a Primary Extract

Set up database credentials to create and run Extract using the steps in Add Database Credentials.
Now, you're ready to add an Extract for your deployment.

1. From the Overview page of the Administration Service, click the **+** sign next to Extracts.
2. Choose the type of Extract to create and click **Next**.

> **Note:**
>
> To learn about creating initial load Extract, see About Instantiating with Initial Load Extract. You can also create a Change Data Capture (CDC) Extract for MySQL and SQL Server databases. See Add a Change Data Capture (CDC) Extract.

3. Provide the required information designated with an asterisk (*). Here's a description of the options in the different sections for the Add Extract screen:

Option	Description	Database
Basic Information Section		
Process Name	Name of the Extract process. The name of the Extract process can be up to 8 characters.	All databases
Description	Description of the Extract process being created.	All databases
Intent	Describes the purpose of creating the Extract. The default option is Unidirectional. Other options are High Availability, Disaster Recovery, N-Way, which are informational only.	All databases
Begin	Used to set the beginning location in the redo or transaction log from which the Extract will start to capture data. Available options are Now, Custom Time, CSN or Position in Log, and EOF depending on the supported database.	All databases
Trail Name	A two character trail name.	All databases
Trail Subdirectory, Size, Sequence, and Offset	You can further configure the trail details.	All databases

Option	Description	Database
Remote	Enable this option if the Extract trail is remote. For Oracle databases, enable this option if the Extract trail is to be written directly to a remote Oracle GoldenGate Classic installation. For MySQL, setting this option enables the `TRANLOGOPTIONS ALTLOGDEST REMOTE` parameter to support a remote Extract, and is not related to trails.	Oracle, MySQL
Registration Information Section		
CSN	Commit Sequence Number (CSN) value	Oracle
Share	Choose the method to share the LogMiner data dictionary. Options are: • Automatic: This option allows the system to choose the method for sharing the dictionary . • None: Choosing this option, will not allow the dictionary to be shared. • Extract: Choose this option to allow sharing the LogMiner dictionary for specific Extract.	Oracle
Optimized	Enable this option to optimize the Extract registration.	Oracle
Downstream Capture	Enable this option to set up a downstream Extract for log mining.	Oracle
Register Only	Use this option to just register the Extract and not add the Extract. The registration creates the replication slot when you register the Extract or use the Register Only option.	PostgreSQL
Source Database Credential		
Create new credential	If you haven't set up your database login credentials, you can create and save the database login credentials from here.	All
Credential Domain	Create a domain for the database.	All
Credential Alias	Specify a credential for the database login.	All
User ID	Specify a user name for logging into the database.	All
Password, Verify Password	Enter the password used to login to the database and reenter the password to verify.	All

Option	Description	Database
Credential Domain	Saves the credential user under the specified domain name. Enables the same alias to be used by multiple Oracle GoldenGate installations that use the same credential store. The default domain is Oracle GoldenGate.	All databases
Credential Alias	Specifies an alias for the user name. Use this option if you do not want the user name to be in a parameter file or command. If `ALIAS` is not used, the alias defaults to the user name, which then must be used in parameter files and commands where a login is required. You can create multiple entries for a user, each with a different alias, by using the `ADD USER` option with `ALIAS`.	All databases
Downstream Mining		
Mining Credential Domain	Domain name of the downstream mining database.	Oracle
Mining Credential Alias	Alias for the mining downstream database.	Oracle
No UserID	Enable this option if there is no source database connection. Selecting this option enables the ADG fetch options.	Oracle
ADG Fetch Credential Domain	Domain name for the ADG fetch database.	Oracle
ADG Fetch Credential Alias	Domain alias for the ADG fetch database.	Oracle

4. (Optional) Enter the encryption profile description. If you have not created an encryption profile, then the Local Wallet profile would be selected, by default.
   a. Select the profile name from the list box. You can select the Local Wallet or a custom profile.
   b. Select the encryption profile type from the list box.
   c. Specify the masterkey for the encryption profile. This option doesn't exist with SQL Server.

5. This is an optional step. Enter the Managed Options while creating all types of Extract processes. See Configure Managed Processes.

   The following table provides these options:

Option	Description
Profile Name	Provides the name of the autostart and autorestart profile. You can select the default or custom options. If you have already created a profile, then you can select that profile also. If you select the Custom option, then you can set up a new profile from this section itself.
Critical to deployment health	(Oracle only) Enable this option if the profile is critical for the deployment health.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.

6. Click **Next**.
7. You can edit the parameter file in the text area to list the table details that you are interested in capturing. For example, `table source.table1;`.
8. You can select Register Extract in the background to register the Extract in the background asynchronously. This option is required for Oracle and PostgreSQL databases. See Register an Extract.
9. Click **Create** and **Run** to create and start the Extract. If you select Create, the Extract is created but you need to start it using the Extract drop-down on the Overview page.

   You are returned to the Overview page of the Administration Service. Select the **Action** list if you want to look at the Extract details such as process information, checkpoint, statistics, parameters, and report.

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 Downstream Database Mining.
- 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.

Add a Change Data Capture (CDC) Extract

These steps configure a CDC Extract to capture transactional data from a source database.

CDC Extract is available with SQL Server and PostgreSQL databases.

> **Note:**
>
> One Extract per database is generally sufficient, but multiple Extracts are allowed if the replication slots are available.

1. Using the Admin Client, or REST API client on the source system, create the Extract parameter file. EDIT PARAMS extname
   In this sample, extname is the name of the primary Extract and matches the name of the Extract that was registered with the database in the previous steps.

   To learn about using Oracle GoldenGate Microservices to perform this task, see Add a Primary Extract.

2. Enter the Extract parameters in the order shown, starting a new line for each parameter statement. Sample basic parameters for Extract for Microservices installations:

```
EXTRACT extname
SOURCEDB dsn_name
USERIDALIAS alias
EXTTRAIL ep
GETTRUNCATES
TABLE schema.*;
```

Parameter	Description
`EXTRACT extname`	`extname` is the name of the Extract and cannot be more than 8 alpha-numeric characters in length. For more information, see extract in Reference for Oracle GoldenGate.
`SOURCEDB dsn_name`	Specifies the name of the database connection DSN.
`USERIDALIAS alias`	Specifies the alias of the database login credential of the user that is assigned to Extract. This credential must exist in the Oracle GoldenGate credential store.
`EXTTRAIL trailname`	Specifies a two character, local trail to which the primary Extract writes captured data.
`GETTRUNCATES`	Optional parameter but needed in order to capture truncation operations.

`TABLE schema.object;`
or
`TABLE schema.*;`
	Specifies the database object for which to capture data.
	• `TABLE` specifies a table or a wildcarded set of tables.
	• `schema` is the schema name or a wildcarded set of schemas.
	• object is the table or sequence name, or a wildcarded set of those objects.
	• * is a wildcard for all tables in the schema.
	Terminate the parameter statement with a semi-colon.
	To exclude a name from a wildcard specification, use the `SCHEMAEXCLUDE`, `TABLEEXCLUDE`, and `EXCLUDEWILDCARDOBJECTSONLY` parameters as appropriate.

> **Note:**
>
> If the schema of tables to be captured from is the same as the schema in `GGSCHEMA` of the `GLOBALS` file, which is not recommended, then you cannot use schema.* in the `TABLE` statement.

3. Enter any optional Extract parameters that are recommended for your configuration. You can edit this file at any point before starting processing by using the `EDIT PARAMS` command.

4. Save and close the file.
5. Add the Extract and its associated trail file.

Topics:

PostgreSQL: Change Data Capture (CDC) Extract

The Oracle GoldenGate Extract process for PostgreSQL receives logical records from the PostgreSQL `test_decoding` database plugin and writes them in commit order into trail files for downstream consumption by a Replicat.

SQL Server: Change Data Capture (CDC) Extract

See CDC Capture Method Operational Considerations for operational considerations when adding a CDC Extract for SQL Server.

Add Online Extract Groups

You can use the MA web interface or the Admin Client command line interface to set up Extract groups in these forms. This section describes the options and parameters used with the `ADD EXTRACT` command.

Topics:

Add an Extract Group

```
ADD EXTRACT group
{, datasource}
{, BEGIN start_point} | {position_point}
[, PARAMS pathname]
[, REPORT pathname]
[, DESC 'description']
```

Where:

- `group` 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 logmining 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.

    `YYYY-MM-DD HH:MM[:SS[.CCCCCC ]]` as 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.
- `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.

Create a Parameter File for Online Extraction

Follow these instructions to create a parameter file for an online Extract group. A parameter file is not required for an alias Extract group.

1. On the source system, issue the following command:
   ```
   EDIT PARAMS name
   ```
   Where:

   `name` is either the name of the Extract group 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.
2. 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` • group is the name of the Extract group that you created with the `ADD EXTRACT` command.	Configures Extract as an online process with checkpoints.
`[SOURCEDB dsn | container | catalog] [, USERIDALIAS alias options | , USERID user, options]`	Specifies database connection information. `SOURCEDB` specifies the source data source name (DSN). See for more information. `USERID` and `USERIDALIAS` specify database credentials if required.

Parameter	Description
RMTHOSTOPTIONS host, MGRPORT port, [, ENCRYPT algorithm KEYNAME key_name]	Specifies the target system, the port where Manager is running, and optional encryption of data across TCP/IP. Only required when sending data over IP to a remote system (if ADD RMTTRAIL was used to create the trail). Not required if the trail is on the local system (if ADD EXTTRAIL was used). Not valid for a passive Extract group.
ENCRYPTTRAIL algorithm	Encrypts all trails that are specified after this entry.
LOGALLSUPCOLS	Use when using integrated Replicat for an Oracle target, or when using Conflict Detection and Resolution (CDR) support. Writes the before images of scheduling columns to the trail. (Scheduling columns are primary key, unique index, and foreign key columns.) See LOGALLSUPCOLS in Reference for Oracle GoldenGate.
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.
SEQUENCE [container.]owner.sequence;	Specifies the fully qualified name of an Oracle sequence to capture. Include the container name if the database is a multitenant container database (CDB).
TABLE [container. | catalog.]owner.object;	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. See Specifying Object Names in Oracle GoldenGate Input for guidelines for specifying object names in parameter files.
CATALOGEXCLUDE SCHEMAEXCLUDE TABLEEXCLUDE EXCLUDEWILDCARDOBJECTSONLY	Parameters that can be used in conjunction with one another to exclude specific objects from a wildcard specification in the associated TABLE statement.

3. Enter any appropriate optional Extract parameters listed in the Oracle GoldenGate Parameters.
4. Save and close the parameter file.

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 **Action** button to start or stop the Extract or view and manage its details. When you select the **Action**, **Details** option 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.

Topics:

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 lost 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.

Start or Stop Extract

From the Administration Service Overview page, click **Action**, **Start/Stop** option. 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.

Delete Extract

To delete an Extract:

1. Stop the Extract using the **Actions**, **Stop** option from the Extract section of the Administration Service Overview page.
2. Click **Delete** to remove the Extract.

> **Note:**
>
> The **Delete** option appears only when the Extract is in **Stopped** state.

Register an Extract

Registering an Extract is required for change synchronization for Oracle and PostgreSQL databases.

Registering 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:srvmgrport 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 code snippet.

   ```
   OGG (https://remotehost:16000oracle_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.

   ```
   OGG (https://remotehost:16000oracle_source)> REGISTER EXTRACT extname
   ```

You can also register an Extract in the background while creating an Extract from the Oracle GoldenGate MA web interface. See Add a Primary Extract for details.

Registering Extract in Microservices Architecture 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. Using the Admin Client, connect to the deployment, then connect to the credential alias for the source database.

```
OGG> CONNECT https://remotehost:srvmgrport 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.

```
OGG (https://remotehost:16000postgresql_source)> REGISTER EXTRACT extname
```

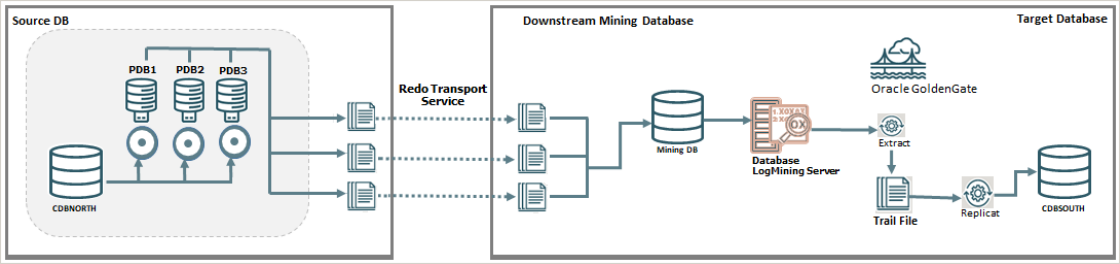
You can also register an Extract from the Oracle GoldenGate MA web interface. See Add a Primary Extract.

Downstream Extract for Downstream Database Mining

Learn about configuring downstream database mining with Oracle GoldenGate Extract and cascaded downstream database mining using Active Data Guard (ADG).

Configure Extract for a Downstream Deployment

A downstream Oracle GoldenGate deployment allows you to offload the source database redo logs to a downstream mining database. A downstream mining database can accept both archived logs and online redo logs from a source database.



This workflow shows the source multitenant container database (CDBNORTH) offloading redo logs to the downstream database (CDBSOUTH) through the logmining server.

> **Note:**
>
> Configuring Extract for a downstream deployment is only applicable to Oracle database.

important for any Data Guard or downstream environment is the setup of the Redo Transport which with the parameter settings of

Evaluate Extract Options for a Downstream Deployment

To configure an Extract on the downstream mining database, consider the following guidelines:

- Multiple source databases can send their redo data to a single downstream database; however the downstream mining database can accept online redo logs from only one of those source databases. The rest of the source databases must ship archived logs.
- When online logs are shipped to the downstream database, real-time capture by Extract is possible. Changes are captured as though Extract is reading from the source logs. In order to accept online redo logs from a source database, the downstream mining database must have standby redo logs configured.
- When using a downstream mining configuration, the source database and mining database must be the same endian and same bit size, which is 64 bits. For example, if the source database was on Linux 64-bit, you can have the mining database run on Windows 64-bit, because they have the same endian and bit size.
- The initialization parameter `db_block_size` must be same between source database and mining database.

Prepare the Source Database for the Downstream Deployment

There must be an Extract user on the source database. Extract uses the credentials of this user to do metadata queries and to fetch column values as needed from the source database.

Add the credentials for connecting Extract to the source database from the Microservices Architecture web interface.

Add Database Credentials to Connect to the Source Database

To create and run Extract and Replicat processes, you need to set up database credentials to connect Extract/Replicat users to the respective source or target databases.

1. Launch the **Administration Service** interface and log in.
2. Click **Configuration** from the **Application Navigation** pane.
3. Click the plus sign (**+**) sign next to Credentials.
4. Enter the following details in the displayed fields:

Database Credential Options	Description
Credential Domain	Specify a domain name to which the database credential is associated. For example, "OracleGoldenGate" is the default domain name, incase you don't specify a domain name.
Credential Alias	This is the alias for your database credential.
User ID	This is the username of the database user. For Oracle database, if you use the EZconnect syntax to connect to the database, then you can specify the value in this field in the following manner: `dbusername@hostname:port/service_name` `dbusername` is the database user name. `hostname` or IP address of the server where the database is running. `port` is the port number for connecting to the database server. Usually, this value is 1521. `service_name` is the name of the service provided in the tnsnames.ora file for the database connection.

Database Credential Options	Description
Password	Password used by database user to log in to the database.

5. Click **Submit**.
6. Click the **Connect to database** icon to test that the connection is working correctly. If the connection is successful, the Connect to database icon turns blue.

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

Configure Redo Transport from Source Database to Downstream Mining Database

To set up the transfer of redo log files from a source database to the downstream mining database, and to prepare the downstream mining database to accept these redo log files, perform the steps given in this topic.

The following summarizes the rules for supporting multiple sources sending redo to a single downstream mining database:

- Only one source database can be configured to send online redo to the standby redo logs at the downstream mining database. The `log_archive_dest_n` setting for this source database should not have a `TEMPLATE` clause.
- Source databases that are not sending online redo to the standby redo logs of the downstream mining database must have a `TEMPLATE` clause specified in the `log_archive_dest_n` parameter.
- Each of the source databases that sends redo to the downstream mining database must have a unique `DBID`. You can select the `DBID` column from the `v$database` view of these source databases to ensure that the DBIDs are unique.
- The `FAL_SERVER` value must be set to the downstream mining database. `FAL_SERVER` specifies the `FAL` (fetch archive log) server for a standby database. The value is a list of Oracle Net service names, which are assumed to be configured properly on the standby database system to point to the desired FAL servers. The list contains the net service name of any database that can potentially ship redo to the downstream database.
- When using redo transport, there could be a delay in processing redo due to network latency. For Extract, this latency is monitored by measuring the delay between LCRs received from source database and reporting it. If the latency exceeds a threshold, a warning message appears in the report file and a subsequent information message appears when the lag drops to normal values. The default value for the threshold is 10 seconds.

Note:

The archived logs shipped from the source databases are called foreign archived logs. You must not use the recovery area at the downstream mining database to store foreign archived logs. Such a configuration is not supported by Extract. Foreign archived logs stored in the Flash Recovery Area (FRA) are not automatically deleted by RMAN jobs. These archived logs must be manually purged.
These instructions take into account the requirements to ship redo from multiple sources, if required. You must configure an Extract process for each of those sources.

To configure redo transport:

1. Configure database connection to connect the source database with the mining database.
2. Configure authentication at each source database and at the downstream mining database to support the transfer of redo data. Redo transport sessions are authenticated using either the Secure Sockets Layer (SSL) protocol or a remote login password file. If a source database has a remote login password file, copy it to the appropriate directory of the mining database system. The password file must be the same at all source databases, and at the mining database.
3. At each source database, configure one `LOG_ARCHIVE_DEST_n` initialization parameter to transmit redo data to the downstream mining database. Set the attributes of this parameter as shown in one of the following examples, depending on whether real-time or archived-log-only capture mode is to be used.
   - Example for real-time capture at the downstream logmining server, where the source database sends its online redo logs to the downstream database:

     ```
     ALTER SYSTEM
     SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC NOREGISTER
     VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)DB_UNIQUE_NAME=dbmscap'
     ```

   - Example for archived-log-only capture at the downstream logmining server:

     ```
     ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=DMBSCAP.EXAMPLE.COM ASYNC
     NOREGISTER VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
     TEMPLATE=/usr/oracle/log_for_dbms1/dbms1_arch_%t_%s_%r.log
     DB_UNIQUE_NAME=dbmscap'
     ```

     Note:

     When using an archived-log-only downstream mining database, you must specify a value for the `TEMPLATE` attribute. Oracle also recommends that you use the `TEMPLATE` clause in the source databases so that the log files from all remote source databases are kept separated from the local database log files, and from each other.

4. At the source database, set a value of `ENABLE` for the `LOG_ARCHIVE_DEST_STATE_n` initialization parameter that corresponds with the `LOG_ARCHIVE_DEST_n` parameter that corresponds to the destination for the downstream mining database, as shown in the following example:

   ```
   ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE
   ```

5. At the source database, and at the downstream mining database, set the `DG_CONFIG` attribute of the `LOG_ARCHIVE_CONFIG` initialization parameter to include the `DB_UNIQUE_NAME` of the source database and the downstream database, as shown in the following example:

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1,dbmscap)'
```

Prepare the Downstream Mining Database to Receive Online Redo Logs

A downstream mining database can accept both archived logs and online redo logs from a source database.

Creating the Downstream Mining User Account

When using a downstream mining configuration, there must be an Extract mining user on the downstream database. The mining Extract process uses the credentials of this user to interact with the downstream logmining server.

The downstream mining user is specified by the `TRANLOGOPTIONS` parameter with the `MININGUSERALIAS` option.

See Add Database Credentials to Connect to the Source Database to assign the correct credentials for the version of your database.

Configure the Mining Database to Archive Local Redo Log Files

This procedure configures the downstream mining database to archive redo data in its online redo logs. These are redo logs that are generated at the downstream mining database.

Archiving must be enabled at the downstream mining database if you want to run Extract in real-time integrated capture mode, but it is also recommended for archive-log-only capture. Extract in integrated capture mode writes state information in the database. Archiving and regular backups will enable you to recover this state information in case there are disk failures or corruption at the downstream mining database.

To Archive Local Redo Log Files:

1. Alter the downstream mining database to be in archive log mode. You can do this by issuing the following

```
 DDL.STARTUP MOUNT;
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
```

2. At the downstream mining database, set the first archive log destination in the `LOG_ARCHIVE_DEST_n` initialization parameter as shown in the following example:

```
ALTER SYSTEM SET
LOG_ARCHIVE_DEST_1='LOCATION=/home/arc_dest/local
VALID_FOR=(ONLINE_LOGFILE,PRIMARY_ROLE)'
```

Alternatively, you can use a command like this example:

```
ALTER SYSTEM SET
LOG_ARCHIVE_DEST_1='LOCATION='USE_DB_RECOVERY_FILE_DEST'
valid_for=(ONLINE_LOGFILE,PRIMARY_ROLE)'
```

> **Note:**
>
> The online redo logs generated by the downstream mining database can be archived to a recovery area. However, you must not use the recovery area of the downstream mining database to stage foreign archived logs or to archive standby redo logs. For information about configuring a fast recovery area, see the Oracle Database Backup and Recovery User's Guide.

3. Enable the local archive destination.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

Configure the Wallet for the Downstream Mining Database

When TDE is enabled on source database and downstream database, then the source wallet or keys should be the same on the downstream mining database and the source database.

Follow these steps to copy the wallet directory from the source database to the downstream mining database:

1. Shutdown the downstream database using the shutdown immediate command.
2. Remove the wallet directory on downstream database view: `rm $T_WORK/wallet/*`
3. Copy the `$T_WORK/wallet/*` from the source database view to the downstream database view.
4. Restart the downstream database.
5. Run checksum on the source database view and downstream database view to ensure that it matches: `cksum $T_WORK/wallet/*`

Prepare a Downstream Mining Database for Real-time Capture

This procedure is only required if you want to use real-time capture at a downstream mining database. It is not required to use archived-log-only capture mode. To use real-time capture, it is assumed that the downstream database has already been configured to archive its local redo data as shown in Configuring the Mining Database to Archive Local Redo Log Files.

Create the Standby Redo Log Files

The following steps outline the procedure for adding standby redo log files to the downstream mining database. The following summarizes the rules for creating the standby redo logs:

- Each standby redo log file must be at least as large as the largest redo log file of the redo source database. For administrative ease, Oracle recommends that all redo log files at source database and the standby redo log files at the downstream mining database be of the same size.
- The standby redo log must have at least one more redo log group than the redo log at the source database, for each redo thread at the source database.

The specific steps and SQL statements that are required to add standby redo log files depend on your environment. See *Oracle Data Guard Concepts and Administration Guide* for detailed instructions about adding standby redo log files to a database.

> **Note:**
>
> If there are multiple source databases sending redo to a single downstream mining database, only one of those sources can send redo to the standby redo logs of the mining database. An Extract process that mines the redo from this source database can run in real-time mode. All other source databases must send only their archived logs to the downstream mining database, and the Extracts that read this data must be configured to run in archived-log-only mode.

To create the standby redo log files:

1. In SQL*Plus, connect to the source database as an administrative user.
2. Determine the size of the source log file. Make note of the results.

   ```
   SELECT BYTES FROM V$LOG;
   ```

3. Determine the number of online log file groups that are configured on the source database. Make note of the results.

   ```
   SELECT COUNT(GROUP#) FROM V$LOG;
   ```

4. Connect to the downstream mining database as an administrative user.
5. Add the standby log file groups to the mining database. The standby log file size must be at least the size of the source log file size. The number of standby log file groups must be at least one more than the number of source online log file groups. This applies to each instance (thread) in a RAC installation. So if you have "n" threads at the source database, each having "m" redo log groups, you should configure n*(m+1) redo log groups at the downstream mining database.

   The following example shows three standby log groups.

   ```
   ALTER DATABASE ADD STANDBY LOGFILE GROUP 3
   ('/oracle/dbs/slog3a.rdo', '/oracle/dbs/slog3b.rdo')
   SIZE 500M; ALTER DATABASE ADD STANDBY LOGFILE
   GROUP 4 ('/oracle/dbs/slog4.rdo', '/oracle/dbs/slog4b.rdo')
   SIZE 500M; ALTER DATABASE ADD STANDBY LOGFILE GROUP 5 ('/oracle/dbs/slog5.rdo', '/
   oracle/dbs/slog5b.rdo') SIZE 500M;
   ```

6. Confirm that the standby log file groups were added successfully.

   ```
   SELECT GROUP#, THREAD#, SEQUENCE#, ARCHIVED, STATUS FROM V$STANDBY_LOG;
   ```

   The output should be similar to the following:

   ```
   GROUP# THREAD# SEQUENCE# ARC STATUS ---------- ---------- ---------- --- ----------
   3 0 0
   YES UNASSIGNED 4 0 0 YES UNASSIGNED 5 0 0 YES UNASSIGNED
   ```

7. Ensure that log files from the source database are appearing in the location that is specified in the

   ```
   LOCATION
   ```

   attribute of the local

   ```
   LOG_ARCHIVE_DEST_n
   ```

that you set. You might need to switch the log file at the source database to see files in the directory.

Configure the Database to Archive Standby Redo Log Files Locally

This procedure configures the downstream mining database to archive the standby redo logs that receive redo data from the online redo logs of the source database. Keep in mind that foreign archived logs should not be archived in the recovery area of the downstream mining database.

To Archive Standby Redo Logs Locally:

1. At the downstream mining database, set the second archive log destination in the `LOG_ARCHIVE_DEST_n` initialization parameter as shown in the following example.

   ```
   ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='LOCATION=/home/arc_dest/srl_dbms1
   VALID_FOR=(STANDBY_LOGFILE,PRIMARY_ROLE)'
   ```

   Oracle recommends that foreign archived logs (logs from remote source databases) be kept separate from local mining database log files, and from each other. You must not use the recovery area of the downstream mining database to stage foreign archived logs.

2. Enable the `LOG_ARCHIVE_DEST_2` parameter you set in the previous step as shown in the following example.

   ```
   ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE
   ```

Enable Downstream Extract to Work with ADG

In a cascaded downstream capture environment, the downstream database does not connect directly to the source database. It uses the Active Data Guard (ADG) as a reference.

Extract must be started using the sourceless option so that it does not connect to the source database and instead connects to ADG using `FETCHUSERID` or `FETCHUSERIDALIAS` when it needs to fetch any non-native datatypes. For example, FETCH operations are processed on the ADG database as this instance is open in read-only mode. Other operations that cannot be processed on the ADG instance, such as creating the dictionary build, are redirected from the ADG to the source database.

When registering a downstream Extract, Oracle GoldenGate connects to ADG as source database instead of the database where the redo originates. ADG redirection is supported for the following commands and parameters:

- `SCHEMATRANDATA`
- `TRANDATA`
- `FLUSH SEQUENCE`
- `TRACETABLE`
- `HEARTBEATTABLE`
- `REGISTER EXTRACT`

> **Note:**
>
> SCHEMATRANDATA and TRANDATA, even though the command is executed on the standby redo log, the actual log groups are created and maintained on the primary database where the actual DML operations take place.

ADG Redirection is available with Oracle Database 21c and higher. It also supports wildcard registration. The following example shows the Extract parameter file for the downstream Extract, when using ADG:

```
EXTRACT EXTDSC
NOUSERID
TRANLOGOPTIONS MININGUSERALIAS cgg_cdbDSC_src DOMAIN OracleGoldenGate
TRANLOGOPTIONS INTEGRATEDPARAMS (DOWNSTREAM_REAL_TIME_MINE Y)
FETCHUSERIDALIAS cgg_cdbADG_src DOMAIN OracleGoldenGate

EXTTRAIL cascade/ea
SOURCECATALOG CDBNORTH_PDB01
DDL INCLUDE MAPPED
TABLE HR.*;
```

Here are the steps to enable downstream Extract to work with ADG Standby:

1. Add an additional LOG_ARCHIVE_DESTINATION_N (LAD) on the ADG standby, as shown in the following example:

   ```
   ALTER SYSTEM SET LOG_ARCHIVE_DEST_3='service=mining_db_service_name ASYNC
   NOREGISTER VALID_FOR(STANDBY_LOGFILES,STANDBY_ROLES)
   DB_UNIQUE_NAME=3rd_db_unique_name' scope=both
   ```

   This step transports and generates the standby_logfiles for an ADG standby.

2. Set the LOG_ARCHIVE_CONFIG on the ADG standby to ship the logs to the mining database, as shown in the following example:

   ```
   ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='dg_config' scope=both;
   ```

   db_config is the database unique name of the first, second, and third databases.

3. On the mining database, set up the location to store the incoming standby_logfiles on the mining database:

   ```
   ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='location= DB_RECOVERY_FILE_DEST
   VALID_FOR=(STANDBY_LOGFILE,ALL_ROLES)' scope=both
   ```

   The location is the database recovery file destination.

4. Run LOG_ARCHIVE_CONFIG on the mining database, so that the Extract process is able to read them on the mining database, as shown in the following example:

   ```
   ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='dg_config' scope=both
   ```

Here, db_config is the database unique name of the first, second, and third databases.

5. For a downstream Extract, you need to ensure that the database connections are appropriately configured. When registering the Extract, make sure that `DBLOGIN` connection is made to the ADG Standby, that is open for read-only activity.
6. To add the Extract and register it, use the following command:

```
DBLOGIN USERID ggadmin@cdbADG_src, PASSWORD ggadmin
MININGDBLOGIN USERID ggadmin@cgg_cdbDSC, password ggadmin
```

`cdbADG_src` is the ADG not primary.

`cgg_cdbDSC` is the mining database.

7. Now, register an Extract that uses the `NOUSERID` parameter:

```
ADD EXTRACT exte, INTEGRATED TRANLOG, BEGIN NOW REGISTER EXTRACT exte
DATABASE
```

8. After the Extract is registered, you can use this Extract to mine data and start the Extract normally.

Use Cases for Downstream Mining Configuration

Read about the different downstream mining configuration use cases.

Case 1: Capture from One Source Database in Real-time Mode

This example captures changes from source database DBMS1 by deploying an Extract at a downstream mining database `DBMSCAP`.

The example assumes that you created the necessary standby redo log files as shown in Configure Extract for a Downstream Deployment.

This assumes that the following users exist:

- User `GGADM1` in DBMS1 whose credentials Extract will use to fetch data and metadata from DBMS1. This user has the alias of ggadm1 in the Oracle GoldenGate credential store and logs in as `ggadm1@dbms1`. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at the source database.
- User `GGADMCAP` in `DBMSCAP` whose credentials Extract will use to retrieve logical change records from the logmining server at the downstream mining database `DBMSCAP`. This user has the alias of ggadmcap in the Oracle GoldenGate credential store and logs in as `ggadmcap@dbmscap`. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at the mining database.

Prepare the Mining Database to Archive its Local Redo

To prepare the mining database to archive its local redo:

1. The downstream mining database must be in archive log mode. You can do this by issuing the following DDL:

```
STARTUP MOUNT;
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
```

2. At the downstream mining database, set `log_archive_dest_1` to archive local redo:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1='LOCATION=/home/arc_dest/local
VALID_FOR=(ONLINE_LOGFILE, PRIMARY_ROLE)'
```

3. Enable `log_archive_dest_1`.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

Prepare the Mining Database to Archive Redo Received in Standby Redo Logs from the Source Database

To prepare the mining database to archive the redo received in standby redo logs from the source database:

1. At the downstream mining database, set log_archive_dest_2 as shown in the following example:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='LOCATION=/home/arc_dest/srl_dbms1
VALID_FOR=(STANDBY_LOGFILE,PRIMARY_ROLE)'
```

2. Enable log_archive_dest_2 as shown in the following example:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

3. Set `DG_CONFIG` at the downstream mining database:

```
 ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1,dbmscap)'
```

Prepare the Source Database to Send Redo to the Mining Database

To prepare the source database to send redo to the mining database::

1. Make sure that the source database is running with the required compatibility:

```
select name, value from v$parameter where name = 'compatible';
```

   The minimum compatibility setting required from integrated capture is 11.1.0.0.0.

2. Set `DG_CONFIG` at the source database:

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1,dbmscap)';
```

3. Set up redo transport at the source database..

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC
OPTIONAL NOREGISTER
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

Set up Extract (ext1) on DBMSCAP

To set up Extract (ext1) on DBMSCAP:

1. Register Extract with the downstream mining database. In the credential store, the alias name of `ggadm1` is linked to a user connect string of `ggadm1@dbms1`. The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

```
DBLOGIN USERIDALIAS ggadm1


MININGDBLOGIN USERIDALIAS ggadmcap


REGISTER EXTRACT ext1 DATABASE
```

2. Create Extract at the downstream mining database:

```
ADD EXTRACT ext1 INTEGRATED TRANLOG BEGIN NOW
```

3. Edit Extract parameter file `ext1.prm`. The following lines must be present to take advantage of real-time capture. In the credential store, the alias name of `ggadm1` is linked to a user connect string of `ggadm1@dbms1`. The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

```
USERIDALIAS ggadm1 TRANLOGOPTIONS MININGUSERALIAS ggadmcap TRANLOGOPTIONS
INTEGRATEDPARAMS (downstream_real_time_mine Y)
```

4. Start Extract.

```
START EXTRACT ext1
```

> **Note:**
>
> You can create multiple Extracts running in real-time Extract mode in the downstream mining database, as long as they all are capturing data from the same source database, such as capturing changes for database DBMS1 in the preceding example.

Case 2: Capture from Multiple Sources in Archive-log-only Mode

The following example captures changes from database `DBMS1` and `DBMS2` by deploying an Extract at a downstream mining database `DBMSCAP`.

It assumes the following users:

- User `GGADM1` in `DBMS1` whose credentials Extract will use to fetch data and metadata from DBMS1. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at `DBMS1`.
- User `GGADM2` in `DBMS2` whose credentials Extract will use to fetch data and metadata from `DBMS2`. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at DBMS2.
- User `GGADMCAP` in `DBMSCAP` whose credentials Extract will use to retrieve logical change records from the logmining server at the downstream mining database. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at the downstream mining database `DBMSCAP`.

This procedure also assumes that the downstream mining database is configured in archive log mode.

Prepare the Mining Database to Archive its Local Redo

To prepare the mining database to archive its local redo:

1. The downstream mining database must be in archive log mode. You can do this by issuing the following DDL.

   ```
   STARTUP MOUNT; ALTER DATABASE ARCHIVELOG; ALTER DATABASE OPEN;
   ```

2. At the downstream mining database, set `log_archive_dest_1` to archive local redo.

   ```
   ALTER SYSTEM SET LOG_ARCHIVE_DEST_1='LOCATION=/home/arc_dest/local
   VALID_FOR=(ONLINE_LOGFILE, PRIMARY_ROLE)'
   ```

3. Enable `log_archive_dest_1`.

   ```
   ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1=ENABLE
   ```

4. Start Extract.

   ```
   START EXTRACT ext1
   ```

> **Note:**
>
> You can create multiple Extracts running in real-time Extract mode in the downstream mining database, as long as they all are capturing data from the same source database, such as capturing changes for database DBMS1 in the preceding example.

Prepare the Mining Database to Archive Redo from the Source Database

Set `DG_CONFIG` at the downstream mining database.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1,dbms2, dbmscap)'
```

Topics:

Prepare the First Source Database to Send Redo to the Mining Database

To prepare the first source database to send redo to the mining database:

1. Make certain that DBMS1 source database is running with the required compatibility.

   ```
   select name, value from v$parameter where name = 'compatible'; NAME VALUE
       --------- -------------------- compatible 11.1.0.0.0
   ```

   The minimum compatibility setting required from capture is 11.1.0.0.0.

2. Set `DG_CONFIG` at DBMS1 source database.

   ```
   ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1, dbmscap)';
   ```

3. Set up redo transport at DBMS1 source database. The `TEMPLATE` clause is mandatory if you want to send redo data directly to foreign archived logs at the downstream mining database.

   ```
   ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC
   OPTIONAL NOREGISTER TEMPLATE='/usr/orcl/arc_dest/dbms1/
   dbms1_arch_%t_%s_%r.log
   VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)DB_UNIQUE_NAME=dbmscap';
   ```

4. Enable the downstream destination.

   ```
   ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
   ```

> **Note:**
>
> You can create multiple Extracts running in real-time Extract mode in the downstream mining database, as long as they all are capturing data from the same source database, such as capturing changes for database DBMS1 in the preceding example.

Prepare the Second Source Database to Send Redo to the Mining Database

To prepare the second source database to send redo to the mining database:

1. Make sure that DBMS2 source database is running with the required compatibility.

   ```
   select name, value from v$parameter where name = 'compatible'; NAME VALUE
       --------- -------------------- compatible 11.1.0.0.0
   ```

The minimum compatibility setting required from capture is 11.1.0.0.0.

2. Set `DG_CONFIG` at DBMS2 source database.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms2, dbmscap)';
```

3. Set up redo transport at DBMS2 source database. The `TEMPLATE` clause is mandatory if you want to send redo data directly to foreign archived logs at the downstream mining database.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC
OPTIONAL NOREGISTER  TEMPLATE='/usr/orcl/arc_dest/dbms2/
dbms2_arch_%t_%s_%r.log
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

> **Note:**
>
> You can create multiple Extracts running in real-time Extract mode in the downstream mining database, as long as they all are capturing data from the same source database, such as capturing changes for database DBMS1 in the preceding example.

Set up Extracts at Downstream Mining Database

These steps set up Extract at the downstream database to capture from the archived logs sent by DBMS1 and DBMS2.

Case 3: Capturing from Multiple Sources with Mixed Real-time and Archive-log-only Mode

The following example captures changes from database DBMS1, DBMS2 and DBMS3 by deploying an Extract at a downstream mining database `DBMSCAP`.

> **Note:**
>
> This example assumes that you created the necessary standby redo log files as shown in Prepare the Downstream Mining Database to Receive Online Redo Logs. It assumes the following users:

- User `GGADM1` in `DBMS1` whose credentials Extract will use to fetch data and metadata from DBMS1. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at `DBMS1`.
- User `GGADM2` in `DBMS2` whose credentials Extract will use to fetch data and metadata from `DBMS2`. It is assumed that the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE()` procedure was called to grant appropriate privileges to this user at DBMS2.

- User GGADM3 in DBMS3 whose credentials Extract will use to fetch data and metadata from DBMS3. It is assumed that the DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE() procedure was called to grant appropriate privileges to this user at DBMS3.
- User GGADMCAP in DBMSCAP whose credentials Extract will use to retrieve logical change records from the logmining server at the downstream mining database. It is assumed that the DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE() procedure was called to grant appropriate privileges to this user at the downstream mining database DBMSCAP.

This procedure also assumes that the downstream mining database is configured in archive log mode.

In this example, the redo sent by DBMS3 will be mined in real time mode, whereas the redo data sent from DBMS1 and DBMS2 will be mined in archive-log-only mode.

Prepare the Mining Database to Archive its Local Redo

To prepare the mining database to archive its local redo:

1. The downstream mining database must be in archive log mode. You can do this by issuing the following DDL:

```
STARTUP MOUNT;
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
```

2. At the downstream mining database, set log_archive_dest_1 to archive local redo.:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1='LOCATION=/home/arc_dest/local
VALID_FOR=(ONLINE_LOGFILE, PRIMARY_ROLE)'
```

3. Enable log_archive_dest_1.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

Prepare the Mining Database to Accept Redo from the Source Databases

Because redo data is being accepted in the standby redo logs of the downstream mining database, the appropriate number of correctly sized standby redo logs must exist. If you did not configure the standby logs, see Create the Standby Redo Log Files.

1. At the downstream mining database, set the second archive log destination in the LOG_ARCHIVE_DEST_n initialization parameter as shown in the following example. This is needed to handle archive standby redo logs.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='LOCATION=/home/arc_dest/srl_dbms3
VALID_FOR=(STANDBY_LOGFILE,PRIMARY_ROLE)'
```

2. Enable the LOG_ARCHIVE_DEST_STATE_2 initialization parameter that corresponds with the LOG_ARCHIVE_DEST_2 parameter as shown in the following example:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

3. Set `DG_CONFIG` at the downstream mining database to accept redo data from all of the source databases.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1, dbms2, dbms3,
dbmscap)'
```

Prepare the First Source Database to Send Redo to the Mining Database

To prepare the first source database to send redo to the mining database:

1. Make certain that DBMS1 source database is running with the required compatibility.

```
select name, value from v$parameter where name = 'compatible';
```

2. Set `DG_CONFIG at DBMS1` source database:

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1, dbmscap)';
```

3. Set up redo transport at `DBMS1` source database. The `TEMPLATE` clause is mandatory if you want to send redo data directly to foreign archived logs at the downstream mining database.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC
OPTIONAL NOREGISTER TEMPLATE='/usr/orcl/arc_dest/dbms1/
dbms1_arch_%t_%s_%r.log
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

Prepare the Second Source Database to Send Redo to the Mining Database

To prepare the second source database to send redo to the mining database::

1. Make sure that DBMS2 source database is running with the required compatibility.

```
select name, value from v$parameter where name = 'compatible';
```

2. Set `DG_CONFIG` at DBMS2 source database:

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms2, dbmscap)';
```

3. Set up redo transport at `DBMS2` source database. The `TEMPLATE` clause is mandatory if you want to send redo data directly to foreign archived logs at the downstream mining database.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC
OPTIONAL NOREGISTER  TEMPLATE='/usr/orcl/arc_dest/dbms2/
dbms2_arch_%t_%s_%r.log
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

Prepare the Second Source Database to Send Redo to the Mining Database

To prepare the second source database to send redo to the mining database::

1. Make sure that DBMS2 source database is running with the required compatibility.

```
select name, value from v$parameter where name = 'compatible';
```

2. Set `DG_CONFIG` at DBMS2 source database:

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms2, dbmscap)';
```

3. Set up redo transport at `DBMS2` source database. The `TEMPLATE` clause is mandatory if you want to send redo data directly to foreign archived logs at the downstream mining database.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC
OPTIONAL NOREGISTER  TEMPLATE='/usr/orcl/arc_dest/dbms2/
dbms2_arch_%t_%s_%r.log
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

Prepare the Third Source Database to Send Redo to the Mining Database

To prepare the third source database to send redo to the mining database:

1. Make sure that DBMS3 source database is running with the required compatibility.

```
select name, value from v$parameter where name = 'compatible'; NAME VALUE
    --------- -------------------- compatible 11.1.0.0.0
```

The minimum compatibility setting required from capture is 11.1.0.0.0.

2. Set `DG_CONFIG` at DBMS3 source database:

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms3, dbmscap)';
```

3. Set up redo transport at DBMS3 source database. Because DBMS3 is the source that will send its online redo logs to the standby redo logs at the downstream mining database, do not specify a `TEMPLATE` clause.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC
    OPTIONAL NOREGISTER
    VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)DB_UNIQUE_NAME=dbmscap';
```

4. Enable the downstream destination:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

Topics:

Set up Extracts at Downstream Mining Database

These steps set up Extract at the downstream database to capture from the archived logs sent by DBMS1 and DBMS2.

Topics:

Set up Extract (ext1) to Capture Changes from Archived Logs Sent by DBMS1

Perform the following steps on the DBMSCAP downstream mining database:

1. Register Extract with `DBMSCAP` for the DBMS1 source database. In the credential store, the alias name of `ggadm1` is linked to a user connect string of `ggadm1@dbms1`.The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`.

```
DBLOGIN USERIDALIAS ggadm1
MININGDBLOGIN
USERIDALIAS ggadmcap
REGISTER EXTRACT ext1 DATABASE
```

2. Add Extract at the mining database `DBMSCAP`:

```
ADD EXTRACT ext1 INTEGRATED TRANLOG BEGIN NOW
```

3. Edit the Extract parameter file `ext1.prm`. In the credential store, the alias name of ggadm1 is linked to a user connect string of `ggadm1@dbms1`. The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`:

```
USERIDALIAS ggadm1 TRANLOGOPTIONS MININGUSERALIAS
ggadmcap TRANLOGOPTIONS INTEGRATEDPARAMS (downstream_real_time_mine N)
```

4. Start Extract:

```
START EXTRACT ext1
```

Set up Extract (ext2) to Capture Changes from Archived Logs Sent by DBMS2

Perform the following steps on the DBMSCAP downstream mining database:

1. Register Extract with the mining database for source database DBMS2. In the credential store, the alias name of `ggadm2` is linked to a user connect string of `ggadm2@dbms2`.The alias name of ggadmcap is linked to a user connect string of `ggadmcap@dbmscap`.

```
DBLOGIN USERIDALIAS ggadm2
MININGDBLOGIN USERIDALIAS ggadmcap
REGISTER EXTRACT ext2 DATABASE
```

2. Create Extract at the mining database:

```
ADD EXTRACT ext2 INTEGRATED TRANLOG, BEGIN NOW
```

3. Edit the Extract parameter file `ext2.prm`. In the credential store, the alias name of ggadm2 is linked to a user connect string of `ggadm2@dbms2`. The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`:

   ```
   USERIDALIAS ggadm2 TRANLOGOPTIONS MININGUSERALIAS ggadmcap TRANLOGOPTIONS
   INTEGRATEDPARAMS (downstream_real_time_mine N)
   ```

4. Start Extract:

   ```
   START EXTRACT ext2
   ```

Set up Extract (ext3) to Capture Changes in Real-time Mode from Online Logs Sent by DBMS3

Perform the following steps on the DBMSCAP downstream mining database:

1. Register Extract with the mining database for source database DBMS3. In the credential store, the alias name of `ggadm3` is linked to a user connect string of `ggadm3@dbms3`.The alias name of ggadmcap is linked to a user connect string of `ggadmcap@dbmscap`.

   ```
   DBLOGIN USERID ggadm3
   MININGDBLOGIN USERID ggadmcap
   REGISTER EXTRACT ext3 DATABASE
   ```

2. Create Extract at the mining database:

   ```
   ADD EXTRACT ext3 INTEGRATED TRANLOG, BEGIN NOW
   ```

3. Edit the Extract parameter file `ext3.prm`. In the credential store, the alias name of ggadm3 is linked to a user connect string of `ggadm3@dbms3`. The alias name of `ggadmcap` is linked to a user connect string of `ggadmcap@dbmscap`:

   ```
   USERIDALIAS ggadm3
   TRANLOGOPTIONS MININGUSERALIAS ggadmcap TRANLOGOPTIONS
   INTEGRATEDPARAMS (downstream_real_time_mine N)
   ```

4. Start Extract:

   ```
   START EXTRACT ext3
   ```

> **Note:**
>
> You can create multiple Extracts running in real-time integrated capture mode in the downstream mining database, as long as they all are capturing data from the same source database, such as all capturing for database DBMS3 in the preceding example.

Positioning Extract to a Specific Start Point

You can position the Extract to a specific start point in the transaction logs using the `ADD/ALTER EXTRACT` commands:

```
{ADD | ALTER EXTRACT} group, LOGNUM log_num, LOGPOS log_pos
```

- *group* is the name of the Oracle GoldenGate Extract group for which the start position is required.
- LOGNUM is the log file number. For example, if the required log file name is test.000034, the LOGNUM value is 34. Extract will search for this log file.

> **Note:**
>
> In Microservices Architecture, ADD EXTRACT will fail if the LOGNUM value contains zeroes preceding the value. For example, ADD EXTRACT ext1, TRANLOG, LOGNUM 000001, LOGPOS 0 will fail. Instead, set LOGNUM to 1 for this example to succeed.

- *LOGPOS* is an event offset value within the log file that identifies a specific transaction record. Event offset values are stored in the header section of a log record. To position at the beginning of a binlog file, set the LOGPOS as 0.

In MySQL logs, an event offset value can be unique only within a given binary file. The combination of the position value and a log number will uniquely identify a transaction record. Maximum Log number length is 8 bytes unsigned integer and Maximum Log offset length is 8 bytes unsigned integer. Log number and Log offset are separated by a pipe ('|') delimiter. Transactional records available after this position within the specified log will be captured by Extract. In addition, you can position an Extract using a timestamp.

Initial Positioning Support for GTID-based Extract for MySQL

MySQL Extract supports initial positioning by GTID set in addition to positioning by timestamp, EOL, and log number or log position. The following MySQL database versions and services support this feature:

- MySQL Server 5.7
- MySQL Server 8.0
- MySQL Database Service (MDS)

In Oracle GoldenGate, 64 KB is the maximum supported size of GTID set as a position by option. In case, the size of the GTID set exceeds the specified size, use the extended checkpoint file to add or alter Extract by GTID set. For more details, see

The syntax of the ADD/ALTER EXTRACT command with position by option as GTIDSET is:

```
{ADD | ALTER EXTRACT} group, TRANLOG, GTIDSET gtid-set
```

The following examples list the use of `GTIDSET` for initial positioning of GTID-based Extract:

```
{ADD | ALTER EXTRACT} group, TRANLOG, GTIDSET "E11FA47-71CA-11E1-9E33-
C80AA9429562:4"


{ADD | ALTER EXTRACT} group, TRANLOG, GTIDSET "3E11FA47-71CA-11E1-9E33-
C80AA9429562:1-10"


{ADD | ALTER EXTRACT} group, TRANLOG, GTIDSET "3E11FA47-71CA-11E1-9E33-
C80AA9429562:1-3:11:47-49"


{ADD | ALTER EXTRACT} group, TRANLOG, GTIDSET "2174B383-5441-11E8-B90A-
C80AA9429562:1-3,24DA167-0C0C-11E8-8442-00059A3C7B00:1-19"
```

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.

7

Instantiate

This section lists details about instantiating with Initial Load Extract and adding the Initial Load Extract using the Admin Client.

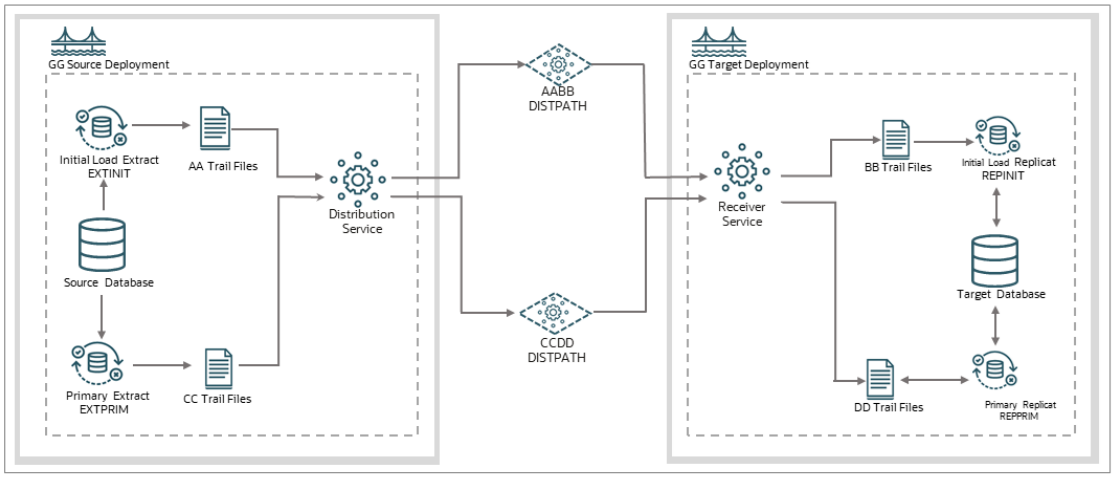
Topics:

About Instantiating with Initial Load Extract

Using the initial load Extract for instantiation, you can replicate data precisely from a source to a target database with zero data loss. To configure this Extract, you'll require a combination of file-based initial load and change data capture (CDC) processes.

In Microservices Architecture, the process of instantiation includes the following tasks:

- Add and configure an Initial Load Extract: This Extract is used to copy the existing contents of one or more tables from the source to the target database.
- Configure Change Data Capture: Used to copy transactional changes from the source to the target database.



> **Note:**
>
> MA doesn't support loading data with an Oracle GoldenGate direct load.

File-based initial load process is the preferred method for performing data replication in MA. It's key components are:

- Initial Load Extract and Replicat: Replicates the existing content of the database tables.
- Primary Extract and Replicat: Replicates change data from the database tables.

- Distribution Paths: Transfers trail files to the target system.

Before you begin, make sure that the database credential alias is created.

You can use the Oracle GoldenGate web interface, Admin Client, or cURL commands to set up this configuration.

Add Initial Load Extract Using the Admin Client

Learn about adding the Initial Load Extract using the Admin Client.

Step 1: Create a Primary Extract

Precise instantiation is used to replicate database resources correctly from the source to the target database. The primary Extract is started first to initiate change data capture early. Precise instantiation is based on the following assumptions:

> **Note:**
>
> For precise instantiation to work, the instantiation SCN must come after the registration SCN.

- The primary Extract is started. It is responsible for change data capture and noting it's registration SCN.
- The database is monitored. The database waits for the oldest open transaction's SCN to come after the registration SCN. This is the instantiation SCN.
- The instantiation SCN is used when creating the initial load Extract and Replicat processes.
- The instantiation SCN is used to create the primary Replicat, once the initial load replication is complete.
- For MySQL, precise instantiation is applicable only for MySQL source and target databases, and is implemented using the `Dump` utility of the MySQL shell. For more information on the `Dump` utility, see MySQL Dump Utility.

To begin, create and start the primary Extract `EXTPRIM` from the AdminClient, as shown in the following example:

Command:

```
OGG (not connected) 1> CONNECT https://oggdep.example.com:9100 as oggadmin
password oggadmin !
```

Output:

```
Using default deployment 'OGGDEP'
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP) 2> DBLOGIN USERIDALIAS oggadmin
```

Output:

```
Successfully logged into database.
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP) 3> ADD EXTRACT extprim INTEGRATED
TRANLOG BEGIN NOW
```

Output:

```
2018-03-16T13:37:07Z INFO OGG-08100 EXTRACT (Integrated) added.
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 4> REGISTER EXTRACT
extprim DATABASE
```

Output:

```
2018-03-16T13:37:30Z INFO OGG-02003 Extract EXTPRIM successfully registered with
database at SCN 1608891.
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 5> EDIT PARAMS extprim
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 6> VIEW PARAMS extprim
```

Output:

```
--
-- E X T P R I M . p r m
-- Primary Extract Parameter File
--
Extract EXTPRIM
UseridAlias oggadmin
ExtTrail AA
Table user01.*;
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 7> ADD EXTTRAIL aa
EXTRACT extprim
```

Output:

```
2018-03-16T13:37:55Z INFO OGG-08100 EXTTRAIL added.
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 8> START EXTRACT extprim
```

Output:

```
2018-03-16T13:38:02Z INFO OGG-00975 EXTRACT EXTPRIM starting
2018-03-16T13:38:02Z INFO OGG-15426 EXTRACT EXTPRIM started
```

In this example, `oggadmin` is the database credential alias.

After creating the primary Extract, retrieve the SCN registration number. Run the `REGISTER EXTRACT` command in the AdminClient. The following example retrieves an SCN value of 1608891.

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 4> REGISTER EXTRACT
extprim DATABASE
```

Output:

```
2018-03-16T13:37:30Z INFO OGG-02003 Extract EXTPRIM successfully registered with
database at SCN 1608891.
```

Step 2: Determine the Instantiation SCN

The Administration Service in Oracle GoldenGate Microservices Architecture, provides an endpoint for retrieving information about open database transactions. This information can be used to identify the SCN to use when instantiating the initial load Extract.

In the following example, the instantiation SCN is 1609723, which is the oldest SCN of all open transactions that is also past the registration SCN of 1608891, identified in the previous step.

```
-- Query for active transactions
--
SELECT T.START_SCN, T.STATUS TSTATUS, T.START_DATE,
       S.SID, S.SERIAL#, S.INST_ID, S.USERNAME, S.OSUSER, S.STATUS SSTATUS,
S.LOGON_TIME
  FROM gv$transaction T
 INNER JOIN gv$session S
 ON s.saddr = t.ses_addr

UNION ALL

--
-- Query for current status
--
SELECT CURRENT_SCN, 'CURRENT', CURRENT_DATE,
       NULL, NULL, NULL, 'SYS', NULL, NULL, NULL
 from v$database

ORDER BY 1;
```

The results of this query can be used to determine the instantiation SCN. The results for this specific query are:

```
1538916 ACTIVE 2018-03-16 18:10:31.0 3865 9176 1 OGGADMIN oracle INACTIVE
2018-03-16 18:10:26.0 1540555 CURRENT 2018-03-16 18:21:50.0 SYS
```

The SCN used to instantiate the initial load Extract is obtained using SQL*Plus. In the following example, the SQL query uses the instantiation SCN value as `1624963`, which is the oldest SCN of all open transactions that are also past the registration SCN of `1608891`.

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 14> SHELL ECHO
'SELECT MIN(START_SCN) FROM gv$transaction;' | ${ORACLE_HOME}/bin/sqlplus -
S / as sysdba

MIN(START_SCN)
--------------
       1624963
```

If there are no open transactions, then this SQL query returns an empty result. A detailed query that takes into account the situation where there are no open transactions is:

```
SELECT MIN(SCN) as INSTANTIATION_SCN
  FROM (SELECT MIN(START_SCN) as SCN
          FROM gv$transaction
         UNION ALL
        SELECT CURRENT_SCN
          FROM gv$database);
```

Step 3: Create and Start the Initial Load Replicat

Before you begin this step, make sure that the checkpoint table `oggadmin.checkpoints`, already exists on the target system. The initial load Replicat is responsible for populating the target database. Run the following command on the AdminClient to create and start the initial load Replicat (`REPINIT`):

Command:

```
OGG (not connected) 1> CONNECT https://oggdep.example.com:9100 as oggadmin
password oggadmin !
```

Output:

```
Using default deployment 'OGGDEP'
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP) 2> DBLOGIN USERIDALIAS oggadmin
```

Output:

```
Successfully logged into database.
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 3> ADD CHECKPOINTTABLE
oggadmin.checkpoints
```

Output:

```
ADD "oggadmin.checkpoints" succeeded.
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 4> ADD REPLICAT repinit
EXTTRAIL dd CHECKPOINTTABLE oggadmin.checkpoints
```

Output:

```
2018-03-16T13:56:41Z INFO OGG-08100 REPLICAT added.
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 5> EDIT PARAMS repinit
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 6> VIEW PARAMS repinit
```

Output:

```
--
--  R E P I N I T . p r m
--  File-Based Initial Load Replicat Parameter File
--
Replicat    REPINIT
UseridAlias oggadmin
Map         user01.*
  Target    user01.*;
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 7> START REPLICAT
repinit
```

Output:

```
2018-03-16T13:58:21Z  INFO    OGG-00975  REPLICAT REPINIT starting
2018-03-16T13:58:21Z  INFO    OGG-15426  REPLICAT REPINIT started
```

Step 4: Create and start the Initial Load Extract

Using the instantiation SCN that you retrieved (`1624963`), the initial load Extract is created to write contents of the database tables to the trail. Create and start the initial load extract, `EXTINIT`.

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 15> ADD EXTRACT extinit
SOURCEISTABLE sourceistable
```

Output:

```
2018-03-16T14:08:38Z INFO OGG-08100 EXTRACT added.
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 16> EDIT PARAMS extinit
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 17> VIEW PARAMS extinit
```

Output:

```
--
--  E X T I N I T . p r m
--  File-Based Initial Load Extract Parameter File
--
Extract     EXTINIT
UseridAlias oggadmin
ExtFile     CC Megabytes 2000 Purge
Table       user01.*, SQLPredicate "As Of SCN 1609723";
```

Command:

```
OGG (https://oggdep.example.com:9100 OGGDEP as oggadmin) 18> START EXTRACT
extinit
```

Output:

```
2018-03-16T14:13:42Z  INFO    OGG-00975  EXTRACT EXTINIT starting
2018-03-16T14:13:42Z  INFO    OGG-15426  EXTRACT EXTINIT started
```

Step 5: Create the Distribution Paths

Create two distribution paths (`AABB` and `CCDD`) for copying the local trails to the remote host from the Admin Client:

Command:

```
OGG (https://oggdep.example.com:9100 oggdep) 15> ADD DISTPATH aabb SOURCE
TRAIL://oggdep.example.com:9102/services/v2/sources?trail=AA target wss://
dallas.oggdevops.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: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 MySQL GTID-based Capture

To use the `gtid` set for precise instantion, make sure that the `gtid_mode` is enabled in the database server and `_DISABLEGTIDRECOVERY true` is not specified in the Extract parameter file. The gtid-based positioning does not work on non-gtid-based capture. The following steps describe how to set up precise instantiation for gtid-based capture:

1. Read the field value of `gtidExecuted` from the `@.json` file and use it as an initial position in the `add extract tranlog gtid set` *`gtid_set`* command.
2. Use this position in Oracle GoldenGate Microservices Admin Client and web interface, as shown in the following example:

```
OGG (https://databasede3phx.oraclevcn.com:9011/ GTIDMAIN) 2> ADD EXTRACT
longgtid, tranlog gtidset "1174b383-3441-11e8-b90a-
c80aa9429920:1-9,\n1174b383-3441-11e8-b90a-c80aa9429921:1-9" 3
```

3. Start the Extract.
4. Verify in the Extract report file for INFO messages for the `Initial GTID set` and `Position of first record processed`. The `Initial GTID set` and `Position of first record processed` should contain the same `gtid` set that is used for the initial positioning.
5. Perform the DML operations and verify that there are no duplicate or missing transactions on the target side.

Precise Instantiation for MySQL non-GTID-based Capture

The following steps describe precise instantiation in the non-gtid based capture:

1. To test the precise instantiation on non-gtid-based capture, make sure either the `gtid_mode` is not enabled in the database server or `_DISABLEGTIDRECOVERY true` is specified in the Extract parameter file.
2. After completing the initial load, use the instance dump.
3. Read the `@.json` file and get the values of `binlogFile` and `binlogPosition` from the `@.json` file. Use these values as `lognum` and `logpos`.
4. Add Extract using `lognum` and `logpos` in Oracle GoldenGate Microservices Architecture, using Admin Client or web interface, as shown in the following example:

```
OGG (https://databasede3phx.oraclevcn.com:9011 GTIDMAIN) 32> ADD EXTRACT
extpos tranlog lognum 5 logpos 1289
```

5. Perform the DML operations and verify that there are no duplicate or missing transactions on the target side. Check the Extract report file for the initial position using `lognum` and `logpos`.

Instantiating for a PostgreSQL Replication 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 intial 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 CHECKTPOINTTABLE 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.

8
Distribute

Learn about the Distribution Service, how to add a distribution path, how to add a target-initiated distribution paths, and managing distribution paths.

About Distribution Service and Distribution Path

The Distribution Service is accessible from the Service Manager home page or you can directly specify the URL in a web browser.

Log in to the Distribution Service for the associated deployment. From the Distribution Service home page you can see a dashboard that displays the path connecting the Extract and Replicat processes. You can add a distribution path or data streams from this interface.

Use the dashboard to perform the following operations.

Action	Reference
• Add distribution paths • Add data streams • View path details • Start or Stop the path • Reposition the path • Enable sharding using filters • Set or customize the DML filtering • Set the DDL filtering • Set or customize Procedure filtering • Customize Tag filtering • Delete a Path	See: #unique_454 Path Actions Also see: `ALTER DISTPATH` command options.

About Distribution Paths

A path is used to send trail data between two data endpoints of a deployment. You can add, monitor, reposition, and manage these paths using the Distribution Service. This topic discusses the steps to create a distribution path (`DISTPATHS`).

A distribution path defines the route for the trail to send and receive data for different topologies. Oracle GoldenGate uses the **target authentication method** to define the method for connecting source and target deployments. The options for setting up the target authentication method are as follows:

- **USER ID ALIAS target authentication**: On the target deployment, a user with Operator role is created and then the credentials of this user are added as credentials in the source Oracle GoldenGate deployment. When using the USERIDALIAS method, while creating the Distribution Path, the value of target authentication method is set to **Password**. The **WSS** (secure web socket) protocol is used for this type of distribution path.
- **Certificate target authentication**: In this case, the distribution path uses trusted CA certificates to access the target deployment. The target authentication method that is set up which creating the Distribution Path is **Certificate**. The **WSS** (secure web socket) protocol is used for this type of distribution path.

- **OAuth target authentication**: In this case, the Oracle GoldenGate user authentication is outsourced to an OAuth service such as IDCS and IAM as cloud-based identity providers and OAM as an on-premise identity provider.

About Distribution 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:

- **USER ID ALIAS target authentication**: On the target deployment, a user with Operator role is created and then the credentials of this user are added as credentials in the source Oracle GoldenGate deployment. When using the USERIDALIAS method, while creating the Distribution Path, the value of target authentication method is set to **Password**. The **WSS** (secure web socket) protocol is used for this type of distribution path.
- **Certificate target authentication**: In this case, the distribution path uses trusted CA certificates to access the target deployment. The target authentication method that is set up which creating the Distribution Path is **Certificate**. The **WSS** (secure web socket) protocol is used for this type of distribution path.
- **OAuth target authentication**: In this case, the Oracle GoldenGate user authentication is outsourced to an OAuth service such as IDCS and IAM as cloud-based identity providers and OAM as an on-premise identity provider.

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.
> - You can also create a user identified with a certificate at the target system. In this case, the source does not require a USERIDALIAS, but the appropriate certificate. See Connecting Two Deployments Using External RootCA Certificate.
> - You cannot use clear text `USERID`/`PASSWORD` credentials for the Distribution Path authorization.

Distribution Path Streaming Protocols

You will need to configure a protocol for the Distribution Path to transfer trail files over the network. This configuration is done when you create a Distribution Path in the Distribution Service.

For details about selecting the streaming protocol, see Add a Distribution Path.

While setting up the Distribution Path, if you select `USERIDALIAS` as the **target authentication method**, then you can select from one of the following protocols that would be used for streaming trail data over the network:

- Secure Web Socket (`wss`): Secure and recommended protocol.
- Web Sockets (`ws`): Unsecure deployments.
- Oracle GoldenGate protocol (`ogg`): Provides interoperability with a non-microservices deployment.

The following matrix provides the combinations of streaming protocols used with Oracle GoldenGate Microservices:

Source/Target	MA Non-Secure	MA Non-Secure with NGINX	MA Secure	Classic Architecture
MA Non-secure	Distribution path with **ws** protocol	Distribution path with **wss** protocol	Distribution path with wss protocol	Distribution Path with **ogg** protocol
MA Non-secure with NGINX	Distribution path with **ws** protocol	Distribution path with **wss** protocol	Distribution path with **wss** protocol	Distribution Path with **ogg** protocol
MA Secure	Distribution path with **ws** protocol	Distribution path with **wss** protocol	Distribution path with **wss** protocol	Distribution Path with **ogg** protocol
Classic Architecture	Oracle GoldenGate pump Extract, connect to the Receiver Service port	Need expose target Receiver Service port, then use Oracle GoldenGate pump Extract to connect to the Receiver port. directly	NA	Regular Oracle GoldenGate pump Extract

Also see, Secure Data in Transit and Configure Reverse Proxy with NGINX to Access Oracle GoldenGate Microservices.

Add a Distribution 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 such as:

- **Path between two secure deployments with USER ID ALIAS target authentication**: On the target deployment, a user with Operator role is created and then the credentials of this user are added as credentials in the source Oracle GoldenGate deployment. The target authentication method that is set up which creating the Distribution Path is "Password". The **wss** (secure web socket) protocol is used for this type of distribution path.
- **Path between two secure deployments with 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 which creating the Distribution Path is "Certificate". The **wss** (secure web socket) protocol is used for this type of distribution path.
- **Path between two deployments with OAuth target authentication**: In this case, the Oracle GoldenGate user authentication is outsourced to an OAuth service such as IDCS.

Paths can also be initiated from the Receiver Service. In cases where there are network security policies that prevent the Distribution Service to open a network connection in the target endpoint to the Receiver Service, the path is initiated from the Receiver Service to the Distribution Service. These types of paths are called target-initiated paths, which are suitable in environments such as Demilitarized Zone Paths (DMZ) or Cloud to on-premise networks.

A path is created to send the trail data over a network from the Extract to the Replicat. To add a distribution path from the Distribution Service:

A path is created to send the transaction of data from the Extract to the Replicat creating a trail path.

1. Log in to the **Distribution Service**.
2. Click the plus (+) sign next to Distribution Paths on the Distribution Service home page. 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.
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.

Source Options Screen	Description
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 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.
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.

Target Options Screen	Description
Alias	UserID alias that has the target Oracle GoldenGate UserID and password.
Trail Options	Enable this option to customize the trail options in the next screen.
Advanced Options	Enable this option 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
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.

Trail Options Screen	Description
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 compresion 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.
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 Enckeys	Enable this option to use an Enckey.
Target Encryption Keyname	The encryption keyname of the Enckeys 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, *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.
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 of the target deployment.

About Target-Initiated Distribution Paths

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.

You can enable this option from the Configuration Assistant wizard Security options, see Add a Deployment.

Add Target-Initiated Distribution Paths

A target-initiated distribution path is created from the Receiver Service. These paths can be used when communication must be initiated from the target.

To create a target-initiated distribution path:

1. Log in to the Receiver Service.
2. Click the + sign on the Overview page to start adding a path.
3. The following table lists the options to set up the path:

Options	Description
Path Name	Name of the target-initiated distribution path
Description	Provide a description of the path.
Reverse Proxy Enabled	Select to use reverse proxy. To know more about configuring your reverse proxy servers, see Configure Reverse Proxy with NGINX to Access Oracle GoldenGate Microservices.
Source Authentication Method	Select the authentication method for the source URI. Authentication options are OAuth 2.0, Certificate, UserID Alias.

Options	Description
Source	From the drop-down list, select your data transfer protocol. The default option is Secure Web Socket Proctocl (wss). Other option is ws. You also need to enter the following details: • Source Host: 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. • 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.
Generated Source URI	A URI is automatically generated for the trail based on the source information you provided.
Target	Name of the target trail of the Replicat you created earlier.
Generated Target URI	A Target URI is automatically generated for the trail based on target trail information you provided.
Target Encryption Algorithm	Select the encryption algorithm for the target trail. Options include AES128, AES192, AES256.
Enable Network Compression	Set the compression threshold value if you decide enable this option.
Sequence Length	The length of the trail sequence number.
Trail Size	The maximum size of a file in a trail.
Configure Trail Format	Toggle this switch to enable and configure the trail file format.
Type	Select one of these types of trail file 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.
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.

Options	Description
Include SYSKEY	Select this option incase your Replicat configuration includes tables with `SYSKEY`.
Quote Style	Select the quote style depending on the database requirements.
Include Column Name?	Enable this option to include column names in the trail file.
Null Is Space?	Select this option to indicate that any null values in the trail file is a space.
Include Place Holder?	Outputs a placeholder for missing columns.
Include Header Fields?	Select to include header fields in the trail file.
Delimiter	An alternative delimiter character.
Use Qualified Name?	Select to use the fully qualified name of the parameter file.
Include Transaction Info?	Enable to to include transaction information.
Encryption Profile	Section
Begin	Select the point from where you need to log data. You can select the following options from the drop-down list: • Now • Custom Time • Position is Log (default)
Source Sequence Number	Select the sequence number of the trail from source deployment Extract.
Source RBA Offset	This setting provides the Relative Byte Address (RBA) offset value which is the point in the trail file (in bytes) from where you want the process to start.
Critical	The default value is false. If set to true, this indicates that the distribution path is critical to the deployment.
Auto Restart	The default value is false. If set to true, the distribution path is restarted automatically when killed.
Auto Restart Options	Set up the auto restart option in this section.
Retries	The number of times to try an restart the task (path process).
Delay	The duration interval to wait between retries.

Role Configuration	Description
Enable filtering	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.

Additional Options	Description
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.
TCP Options	Section

Additional Options	Description
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.

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 Distrbution 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`.

Manage Distribution Paths

Learn about managing distribution paths.

Topics:

Path Actions

After a new path is added, you can perform actions such as stop or pause a path, view reports and statistics, reposition the path, change its filtering, and delete a path.

On the Overview page of the Distribution Service, click the **Action** button next the Distribution Path name. From the drop-down list, you can use the following path actions:

- **Details**: Use this option to view details of the path. You can view the path information including the source and target. You can also edit the description of the path. Statistical data is also displayed including 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 these 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 Distrbution Service.
- **Delete**: Use this 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
- **Change Filtering**: Use this option to enter sharding, DML filtering, DDL filtering, Procedure filtering, and Tag filtering options.

Depending on the action you select, you can see the change in status at the bottom of the Overview page.

Reposition a Path

You can reposition a path as required. To reposition a distribution path or target-initiated distribution path:

1. From the Distribution Service home page, click **Distribution Path** to open the Distribution Paths page.
2. Click the **Action** button for the path and select **Reposition** from the drop-down list. The Reposition dialog box is displayed.
3. Specify the source trail Sequence Number and the Source RBA Offset.
4. Click **Apply**.

Change the Path Filtering

If you want to change the filter settings for an existing path, the steps are mostly the same as those for creating the filtering for a new path.

From the left navigation pane of the Distribution Service home page, click Distribution Path.

For the specific distribution path, click **Action**. From the drop-down list, click **Change Filtering**.

Rule Cofiguration	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 Cofiguration	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*.

Review the Distribution Path Information

You can constantly monitor the activity of the path on the Distribution Service Process Information page.

- The path details that you configured. You can change the Description, source and target URIs, Target Authentication Method, DB Name, Target Encryption Algorithm, Enable Network Compression, Sequence Length, Trail Size, configure trail format, mark as Critical and enable Auto Restart. When changing the trail format, be sure to apply your changes.
- The advanced options are the delay, flush, and TCP that you configured. You can change any or all of these options, then apply to the path.

The **Statistics** tab shows you detailed information about the path, such as the different path types and tables. You can use the arrows to sort the tables and the search to quickly locate a specific table. The search is case insensitive and starts searching as you type to update the table.

9
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.

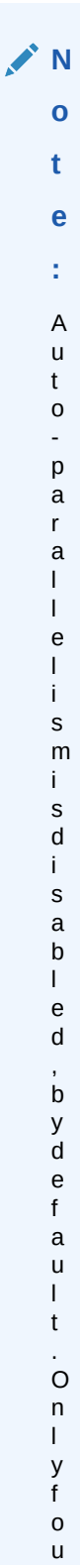
Select a Replicat Type for your Deployment

The Replicat process 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



Note:

Auto-parallelism is disabled, by default. Only you u

Feature	Parallel Replicat	Integrated Replicat	Coordinated Replicat	Classic Replicat
	r t h r e a d s a r e u s e d i n t h e d e f a u l t s e t t i n g s . I f y o u w a n t t o c h a n g e R e			

Feature	Parallel Replicat	Integrated Replicat	Coordinated Replicat	Classic Replicat
	plicatt ouse MIN _PARALLELISM and MAX _PARALLELISM, thenaut			

Feature	Parallel Replicat	Integrated Replicat	Coordinated Replicat	Classic Replicat
	o-parallelism is 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

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 logmining 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 any implementation; Oracle or heterogeneous database configurations.

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. The specified user is subject to the rules of the `GETREPLICATES` or `IGNOREREPLICATES` parameter.

Additional Parameter Options for Integrated Replicat

You can set these parameters by using the `DBOPTIONS` parameter with the `INTEGRATEDPARAMS` option or dynamically by issuing the `SEND REPLICAT` command with the `INTEGRATEDPARAMS` option.

The default Replicat configuration should be sufficient. However, if needed, you can set the following inbound server parameters to support specific requirements.

> **Note:**
>
> For detailed information and usage guidance for these parameters, see the "`DBMS_APPLY_ADM`" section in *Oracle Database PL/SQL Packages and Types Reference*.
>
> See `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_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).

Types of Replicat

The Replicat process can be configured in the following three modes (also referred to as Replicat types):

- **Classic Replicat**: In classic mode, Replicat is a single-threaded process that uses standard SQL to apply data to the target tables. See Classic Replicat for more details.
- **Coordinated Replicat**: In this mode, the Replicat process is threaded. One coordinator thread spawns and coordinates one or more threads that execute replicated SQL operations in parallel. A coordinated Replicat process uses one parameter file and is monitored and managed as one unit. See Coordinated Replicat for more details.
- **Integrated Replicat**: In this mode, the Replicat process leverages the apply processing functionality that is available within the Oracle Database. Within a single Replicat configuration, multiple inbound server child processes known as apply servers apply transactions in parallel while preserving the original transaction atomicity. See About Integrated Replicat for more details.
- **Parallel Replicat**: Is a new variant of Replicat that applies transactions in parallel to improve performance. Parallel Replicat only supports replicating data from trails with full metadata, which requires the classic trail format. It takes into account dependencies between transactions, similar to Integrated Replicat. See Parallel Replicat for more details. Parallel Replicat is available in non-integrated (classic) and integrated mode.
- **Initial Load Replicat**: In this mode, when you set up Oracle GoldenGate for initial loads, the Replicat process applies a static data copy to target objects or routes the data to a high-speed bulk-load utility. See Add Initial Load Extract Using the Admin Client for more details.

About Classic or Non-Integrated Replicat

In classic mode, Replicat is a single-threaded process that uses standard SQL to apply data to the target tables. In this mode, Replicat operates as follows:

- Reads the Oracle GoldenGate trail.
- Performs data filtering, mapping, and conversion.
- Constructs SQL statements that represent source database DML or DDL transactions (in committed order).
- Applies the SQL to the target through the SQL interface that is supported for the given target database, such as ODBC or the native database interface.

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 Integrated Replicat

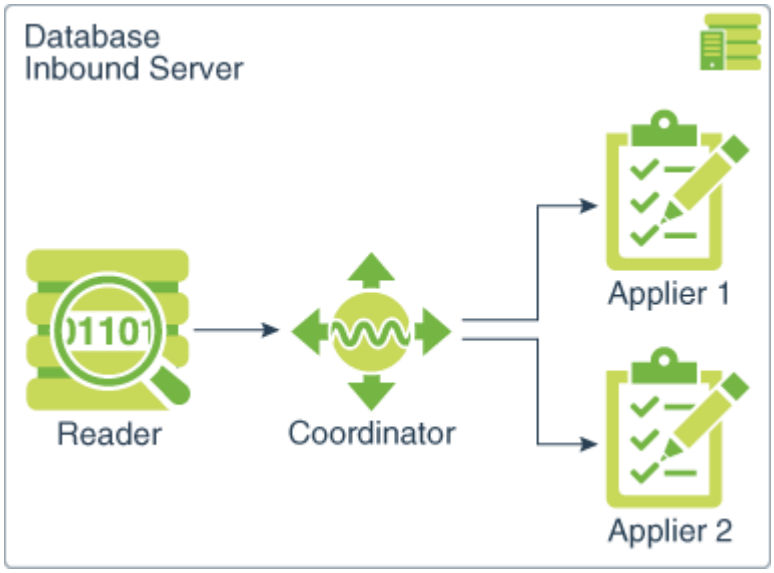
In integrated mode, Replicat leverages the apply processing functionality that is available within the target Oracle database. In this mode, Replicat reads the trail, constructs logical change records that represent source DML or DDL transactions, and transmits these records to an inbound server in the 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
- `UDT`

> **Note:**
>
> By default, UDT's are applied with the inbound server. Only if `NOUSENATIVEOBJSUPPORT` is in place, then Extract handling is done by Replicat directly.

Because transactions are applied serially, heavy use of such operations may reduce the performance of the integrated Replicat mode. Integrated Replicat performs best when most of the apply processing can be performed in integrated mode.

> **Note:**
>
> User exits are executed in integrated mode. However, user exit may produce unexpected results, if the exit code depends on data in the replication stream.

> **Note:**
>
> Integrated Replicat requires that any foreign key columns are indexed.

Benefits of Integrated Replicat

The following are the benefits of using integrated Replicat versus non-integrated Replicat.

- Integrated Replicat enables heavy workloads to be partitioned automatically among parallel apply processes that apply multiple transactions concurrently, while preserving the integrity and atomicity of the source transaction. Both a minimum and maximum number of apply processes can be configured with the `PARALLELISM` and `MAX_PARALLELISM` parameters. Replicat automatically adds additional servers when the workload increases, and then adjusts downward again when the workload lightens.
- Integrated Replicat requires minimal work to configure. All work is configured within one Replicat parameter file, without configuring range partitions.
- High-performance apply streaming is enabled for integrated Replicat by means of a lightweight application programming interface (API) between Replicat and the inbound server.
- Barrier transactions are coordinated by integrated Replicat among multiple server apply processes.
- DDL operations are processed as direct transactions that force a barrier by waiting for server processing to complete before the DDL execution.
- Transient duplicate primary key updates are handled by integrated Replicat in a seamless manner.

Integrated Replicat Requirements

To use integrated Replicat, the following must be true.

- Supplemental logging must be enabled on the source database to support the computation of dependencies among tables and scheduling of concurrent transactions on the target. Instructions for enabling the required logging are in Configuring Logging Properties. This logging can be enabled at any time up to, but before you start the Oracle GoldenGate processes.
- Integrated Parallel Replicat is supported on Oracle Database 12.2.0.1 and greater.

About 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.

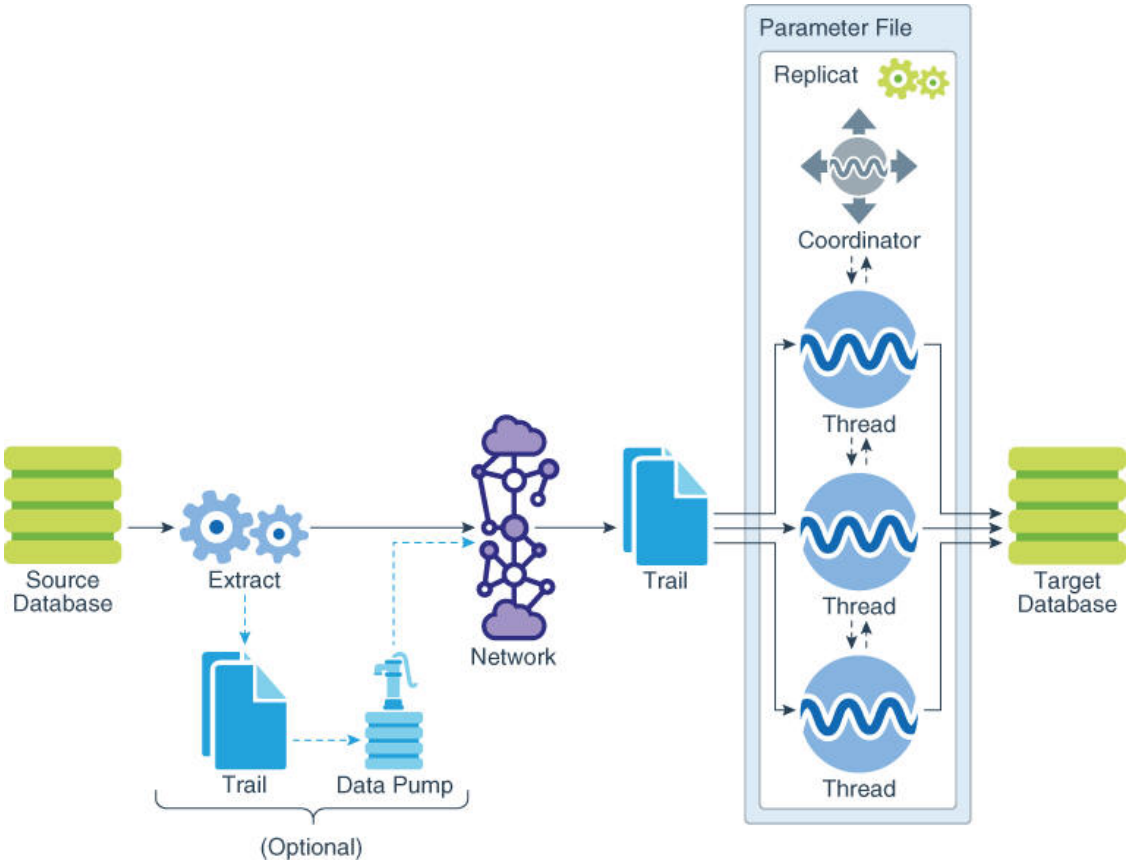
The difference between classic mode and coordinated mode is that Replicat is multi-threaded in coordinated mode. Within a single Replicat instance, multiple threads read the trail independently and apply transactions in parallel. Each thread handles the filtering, mapping, conversion, SQL construction, 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 Parallel Replicat

Parallel Replicat is another variant of Replicat that applies transactions in parallel to improve performance.

It takes into account dependencies between transactions, similar to Integrated Replicat. The dependency computation, parallelism of the mapping and apply is performed outside the database so can be off-loaded to another server. The transaction integrity is maintained in this process. In addition, parallel Replicat supports the parallel apply of large transactions by splitting a large transaction into chunks and applying them in parallel.

Parallel Replicat supports the following two modes: Integrated and Non-integrated. Only Oracle database supports parallel Replicat and integrated parallel Replicat. However, parallel Replicat supports all databases when using the non-integrated option.

To use parallel Replicat, you need to ensure that you have the following values, which are also the default values:

- Metadata in the trail (which means you can't use parallel Replicat if your trails are formatted below 12.1).
- You must have scheduling columns in your trail file.
- You must use UPDATERCORDFORMAT COMPACT.

With integrated parallel Replicat, the Replicat sends the LCRs to the inbound server, which applies the data to the target database, and in regular parallel Replicat, Oracle GoldenGate applies the LCR as a SQL statement directly to the database, similar to how the other non-integrated Replicats work.

> **Note:**
>
> For best performance for an OLTP workload, parallel Replicat in non-integrated mode is recommended.

The components of parallel Replicat are:

- Mappers operate in parallel to read the trail, map trail records, convert the mapped records to the Integrated Replicat LCR format, and send the LCRs to the Merger for further processing. While one Mapper maps one set of transactions, the next Mapper maps the next set of transactions. The trail information is split and the trail file is untouched because it orders trail information in order.
- Master processes have two threads, Collater and Scheduler. The Collater receives mapped transactions from the Mappers and puts them back into trail order for dependency calculation. The Scheduler calculates dependencies between transactions, groups transactions into independent batches, and sends the batches to the Appliers to be applied to the target database.
- Appliers reorder records within a batch for array execution. It applies the batch to the target database and performs error handling. It also tracks applied transactions in checkpoint tables.

> **Note:**
>
> Parallel Replicat requires that any foreign key columns are indexed.

Benefits of Parallel Replicat

The following are the benefits of using parallel Replicat:

- Integrated Parallel Replicat enables heavy workloads to be partitioned automatically among parallel apply processes that apply multiple transactions concurrently, while preserving the integrity and atomicity of the source transaction. Both a minimum and maximum number of apply processes can be configured with the `PARALLELISM` and

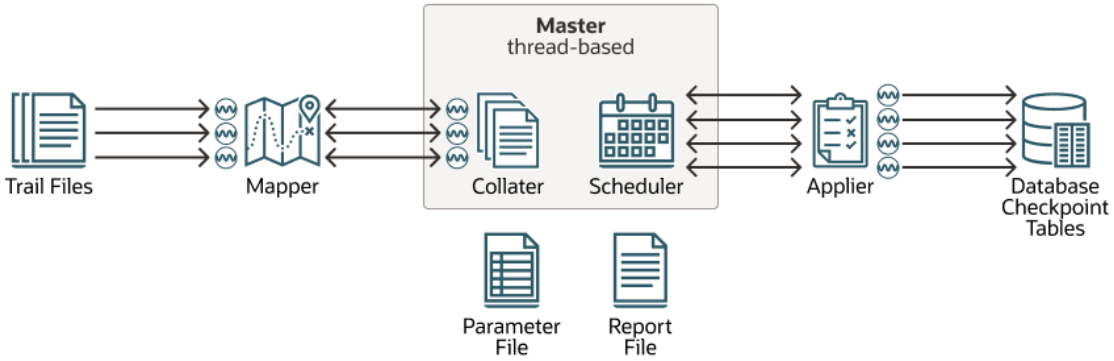
MAX_PARALLELISM parameters. Replicat automatically adds additional servers when the workload increases, and then adjusts downward again when the workload lightens.

- Integrated Parallel Replicat requires minimal work to configure. All work is configured within one Replicat parameter file, without configuring range partitions.
- High-performance apply streaming is enabled for integrated parallel Replicat by means of a lightweight application programming interface (API) between Replicat and the inbound server.
- Barrier transactions are coordinated by integrated parallel Replicat among multiple server apply processes.
- DDL operations are processed as direct transactions that force a barrier by waiting for server processing to complete before the DDL execution.
- Transient duplicate primary key updates are handled by integrated parallel Replicat in a seamless manner.
- Parallel Replicat can break a single large transaction into smaller chunks and apply those chunks in parallel. See SPLIT_TRANS_RECS for details.

Parallel Replication Architecture

Parallel replication processes leverage the apply processing functionality that is available within the Oracle Database in integrated mode. Within a single Replicat configuration, multiple inbound server child processes, known as apply servers, apply transactions in parallel while preserving the original transaction atomicity.

The following architecture diagram depicts the flow of change records through the various processes of a parallel replication from the trail files to the target database, for a non-integrated parallel Replicat.



The following is the description of the architecture diagram given above:

- The Mappers read the trail file and map records, forward the mapped records to the Master. The batches are sent to the Appliers where they are applied to the target database.
- The Master process consists of two separate threads, Collater and Scheduler. The Collater is responsible for managing and communicating with the Mappers, along with receiving the mapped transactions and reordering them into a single in-order stream. The Scheduler is responsible for managing and communicating with the Appliers, along with reading transactions from the Collater, batching them, and scheduling them to Appliers.
- The Scheduler controller communicates with the Scheduler to gather any necessary information (such as, the current low watermark position). The Scheduler controller is

required for CDB mode for Oracle Database because it is responsible for aggregating information pertaining to the different target PDBs and reporting a unified picture. The Scheduler controller is created for simplicity and uniformity of implementation, even when not in CDB mode. Every process reads the parameter file and shares a single checkpoint file.

Basic Parameters for Parallel Replicat

The following table lists the basic parallel Replicat parameters and their description.

Parameter	Description
MAP_PARALLELISM	Configures number of mappers. This controls the number of threads used to read the trail file. The minimum value is 1, maximum value is 100 and the default value is 2.
APPLY_PARALLELISM	Configures number of appliers. This controls the number of connections in the target database used to apply the changes. The default value is four.
MIN_APPLY_PARALLELISM MAX_APPLY_PARALLELISM	The Apply parallelism is auto-tuned. You can set a minimum and maximum value to define the ranges in which the Replicat automatically adjusts its parallelism. There are no defaults. Do not use with APPLY_PARALLELISM at same time.
SPLIT_TRANS_REC	Specifies that large transactions should be broken into pieces of specified size and applied in parallel. Dependencies between pieces are still honored. Disabled by default.
COMMIT_SERIALIZATION	Enables commit FULL serialization mode, which forces transactions to be committed in trail order.
Advanced Parameters	
LOOK_AHEAD_TRANSACTIONS	Controls how far ahead the Scheduler looks when batching transactions. The default value is 10000.
CHUNK_SIZE	Controls how large a transaction must be for parallel Replicat to consider it as large. When parallel Replicat encounters a transaction larger than this size, it will serialize it, resulting in decreased performance. However, increasing this value will also increase the amount of memory consumed by parallel Replicat.

Example Parameter File

```
REPLICAT repe USERID ggadmin, password ***
MAP_PARALLELISM 3
MIN_APPLY_PARALLELISM 2
MAX_APPLY_PARALLELISM 10
SPLIT_TRANS_RECS 60000
MAP *.*, TARGET *.*;
```

Add a Replicat

Learn about the prerequisites to add a Replicat, select the approriate Replicat type, and the steps to add a Replicat.

Before you Add a Replicat

Before you add a Replicat, add a checkpoint table. After you connect to the database, you can create the checkpoint table by following these steps:

1. From the Administration Service, go the Configuration page using the navigation pane.
2. Click the + sign next to the **Checkpoint** section on the **Database** tab.
3. Enter the checkpoint table name in the **Checkpoint Table** box. The table name must be a two-part or three-part value. For example, `ggadmin.ggs_checkpointtable`.

You can add the checkpoint table using the `ADD CHECKPOINTTABLE` command from the Admin Client.

Add a Checkpoint Table

Not valid for Replicat for Java, Oracle GoldenGate Applications Adapter, or Oracle GoldenGate for Big Data.

Use `ADD CHECKPOINTTABLE` to create a checkpoint table in the target database. Replicat uses the table to maintain a record of its read position in the trail for recovery purposes.

The use of a checkpoint table is strongly recommended, because it causes checkpoints to be part of the Replicat transaction. This allows Replicat to recover more easily in certain circumstances than when a checkpoint file alone is used. Parallel and Coordinated Replicats require checkpoint tables.

One table can serve as the default checkpoint table for all Replicat groups in an Oracle GoldenGate instance if you specify it with the `CHECKPOINTTABLE` parameter in a `GLOBALS` file. More than one instance of Oracle GoldenGate (multiple installations) can use the same checkpoint table. Oracle GoldenGate keeps track of the checkpoints even when the same Replicat group name exists in different instances.

Use the `DBLOGIN` command to establish a database connection before using this command. Do not change the names or attributes of the columns in this table. You may, however, change table storage attributes.

Admin Client Syntax

```
ADD CHECKPOINTTABLE [[container. | catalog.] owner.table]
```

The name cannot contain any special characters, such as quotes, backslash, dollar sign, and percent symbol. Record the name of the table, because you need it to view statistics or delete the table if needed.

The owner and name can be omitted if you are using this table as the default checkpoint table and it is listed with `CHECKPOINTTABLE` in the `GLOBALS` file. It is recommended, but not required, that the table be created in a schema dedicated to Oracle GoldenGate. If an owner and name are not specified, a default table is created based on the `CHECKPOINTTABLE` parameter in the `GLOBALS` parameter file.

Record the name of the table, because you will need it to view statistics or delete the table if needed.

Record the name of the checkpoint table as that will be used when you add a Replicat, or delete a Replicat and need to drop the checkpoint table using the `DELETE CHECKPOINTTABLE` command.

The default schema for the checkpoint table is controlled by the Oracle GoldenGate user that is defined for each deployment.

Examples

The following adds a checkpoint table with the default name specified in the `GLOBALS` file.

```
ADD CHECKPOINTTABLE
```

The following adds a checkpoint table with a user-defined name.

```
ADD CHECKPOINTTABLE ggadmin.ggs_checkpointtable
```

Add a Replicat

You can add Replicats for the target deployment from the Administration Service. Make sure that you have configured your deployments correctly, checked your database credentials, and created an Extract before you set up your Replicat. For details, see First Access to the Deployment from the Service Manager. Once you've set up your source and target deployment, you can create and run the Replicat by following these steps:

1. Click the + sign next to Replicats on the Administration Service home page. The Add Replicat page is displayed.
2. Select a Replicat type and click **Next**.

> **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:

   - Integrated Replicat
   - Non-integrated Replicat: This option is displayed with heterogeneous or non-Oracle databases.
   - Classic Replicat: This option is displayed with Oracle database.
   - Coordinated Replicat
   - 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.
3. Enter the required Replicat options on the Replicat Options page and click **Next**. To know more about the Replicat options, see the online help.
4. For managed processes, the options to enter are:

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.

5. Check the Replicat parameter files and modify it as follows:

```
REPLICAT repe
USERIDALIAS ggwest DOMAIN OracleGoldenGate
--
DDL EXCLUDE ALL
DDLERROR default discard
REPERROR (default,discard)
DDLOPTIONS REPORT
SOURCECATALOG DBEAST MAP hr.*, TARGET hr.*;
```

6. Click **Create and Run** to create and run the Replicat.

Basic Parameters for Different Replicat Modes

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.

   Basic parameters for the Replicat group in non-integrated mode:

   ```
   REPLICAT repe
   USERIDALIAS ggeast
   MAP hr.*, TARGET hr.*;
   ```

   Basic parameters for the parallel Replicat in integrated mode:

   ```
   REPLICAT repe
   USERIDALIAS ggeast
   DBOPTIONS INTEGRATEDPARAMS
   MAP hr.*, TARGET hr.*;
   ```

Parameter	Description
`REPLICAT group`	`group` is the name of the Replicat group.
`USERIDALIAS alias`	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 DEFERREFCONST`	Applies to Replicat in nonintegrated mode. `DEFERREFCONST` sets constraints to `DEFERRABLE` to delay the enforcement of cascade constraints by the target database until the Replicat transaction is committed.
`DBOPTIONS INTEGRATEDPARAMS (parameter[, ...])`	This parameter specification applies to Replicat in integrated mode. It specifies optional parameters for the inbound server.
`MAP [container.]schema.object, TARGET schema.object;`	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. Terminate this parameter statement with a semi-colon. To exclude objects from a wildcard specification, use the `MAPEXCLUDE` parameter.

3. If using integrated Replicat or parallel Replicat in integrated mode, add the following parameters to the Extract parameter file:

- `LOGALLSUPCOLS`: This is a default option, which is preset for the Replicat parameter. This parameter ensures the capture of the supplementally logged columns in the before image. It's the default parameter and shouldn't be turned off or disabled. It is valid for any source database that is supported by Oracle GoldenGate. For Extract versions older than 12c, you can use `GETUPDATEBEFORES` and `NOCOMPRESSDELETES` parameters to satisfy the same requirement. The database must be configured to log the before and after values of the primary key, unique indexes, and foreign keys.
- The `UPDATERECORDFORMAT` parameter set to `COMPACT`: This is a default option, which is preset for the Replicat parameter. This setting causes Extract to combine the before and after images of an `UPDATE` operation into a single record in the trail. This is the default option and it is recommended that you don't change the default setting.

4. 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.
5. Save and close the file.

Replicat Actions

Replicat actions include starting or stopping Replicat, alter Replicat parameters, forcibly stopping Replicat, or deleting a Replicat.

You can manage a Replicat process or a Replicat group from the Administration Service home page.

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

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 displayd 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.

10
Secure

This section lists details about creating an applying master and encryption keys, streaming protocols, managing identities, and configuring Kerberos authentication.

Topics:

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.

Supported	Oracle GoldenGate Security Feature	Description
✔	Secure Administration	• All administrative operations are authorized and made with REST API calls • Support for TLS 1.3 (default) and TLS 1.2
✔	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
✔	Password management and MFA	• Local password policy based on character length and rules • Supported and enforced by external Identity Provider (IDP)
✔	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.

Supported	Oracle GoldenGate Security Feature	Description
✔	Database Credentials	• Credentials stored in secure PKCS#12 wallet • Kerberos support, if available for the database, for example: Oracle.
✔	Database Connectivity	Support based on Database client capabilities Example: Oracle using TCPS or Native Network Encryption.
✔	Network Trail File Distribution (Data in Transit)	• Secured with industry standard secure streaming Websocket protocol (WSS) • Support for mutual TLS using client certificate
✔	Proxy/DMZ	• Support for reverse and forward Proxy • Target-initiated Distribution Path for DMZ systems
✔	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
✔	Auditing	• Logging of REST API Calls • Enabled System Security Logging

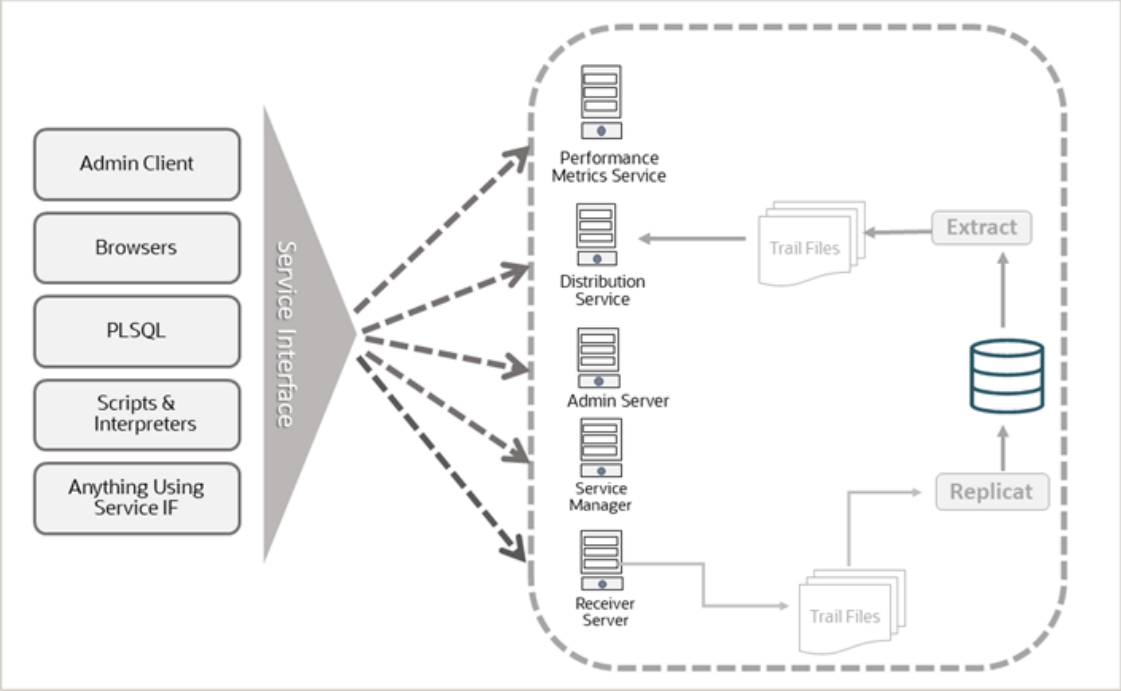
Secure Deployments

Oracle GoldenGate deployment can be secured in two ways:

1. Secure an Oracle GoldeGate deployment with Server Certtificates. 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 Reverse Proxy with NGINX to Access Oracle GoldenGate Microservices.

> **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.
- 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.

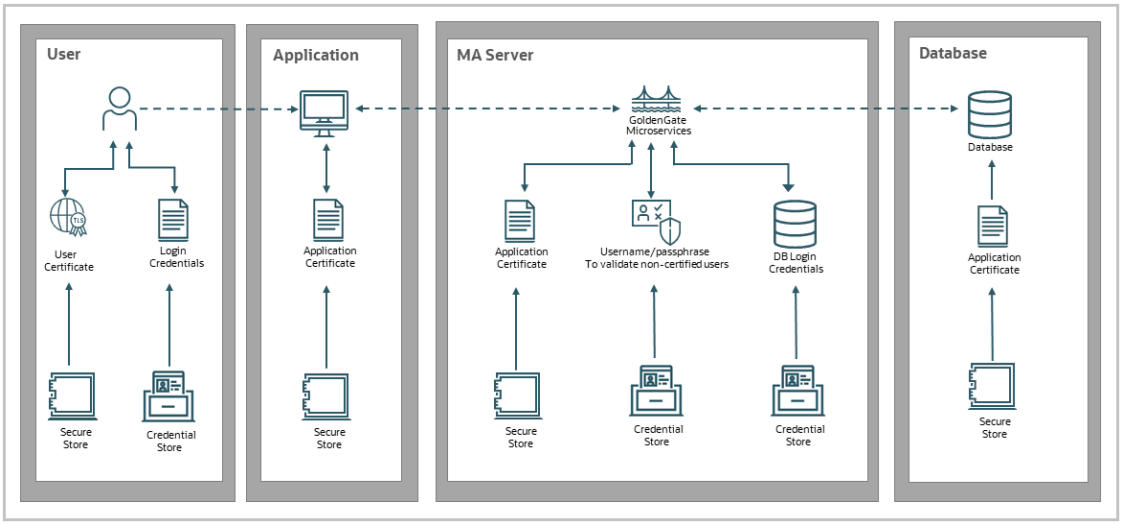
Authorization (AZ) includes Oracle GoldenGate to Oracle GoldenGate communication, and tasks for network and server configuration.

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.

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.

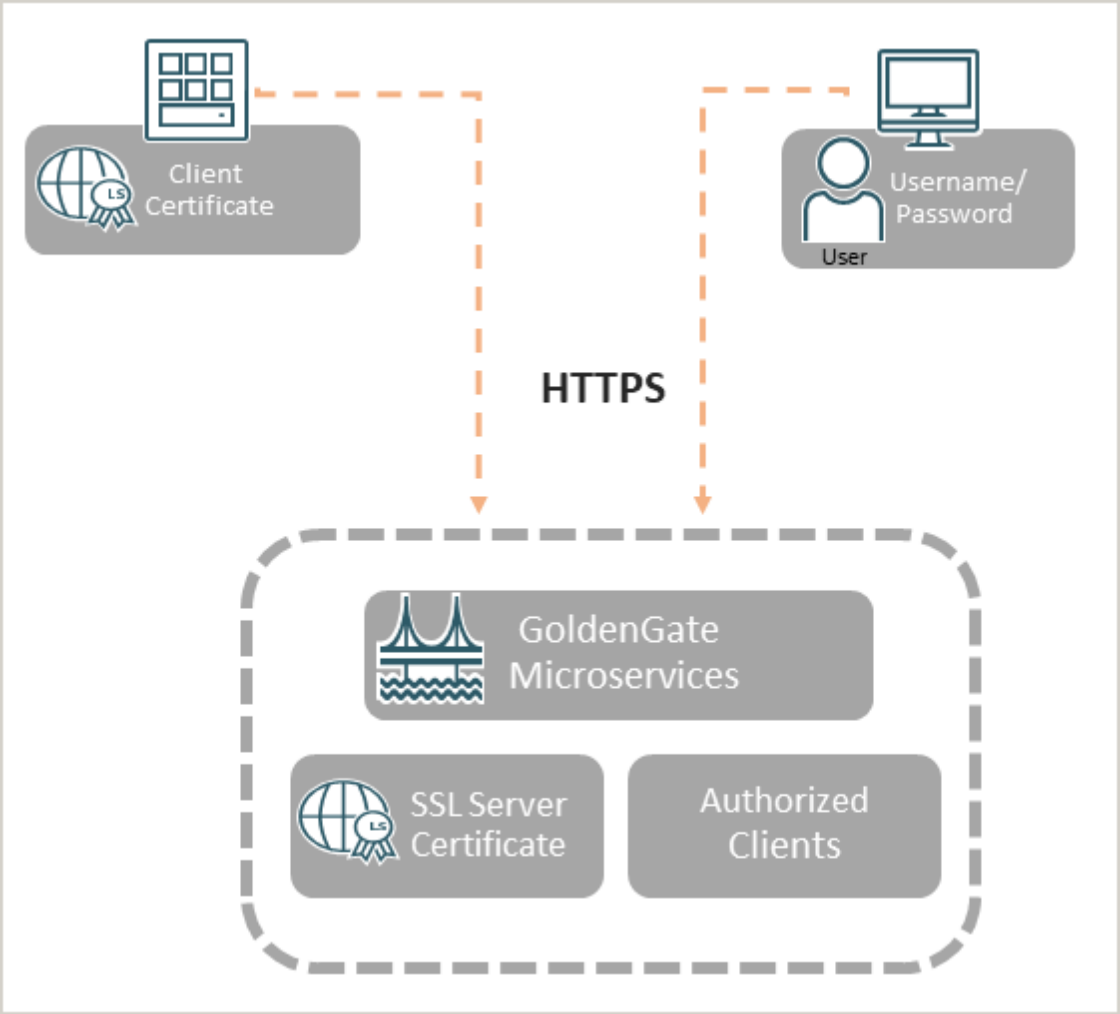
You can set up Certificate authentication when configuring a distribution path to connect the source deployment to a target deployment that is on a different network. This scenario is most common different organizations need to communicate over a secure network for data transfer. In this case, a distribution 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.

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 a 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

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 10-1 Oracle GoldenGate User Roles and Privileges

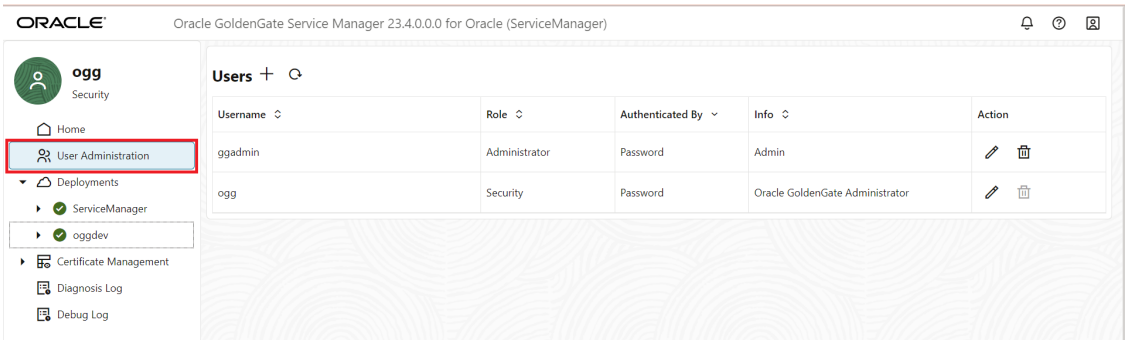
Role ID	Privilege Level
User	Allows information-only service requests, which do not alter or effect the operation of either the MA. Examples of Query/Read-Only information include performance metric information and resource status and monitoring information.
Operator	Allows users to perform only operational 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

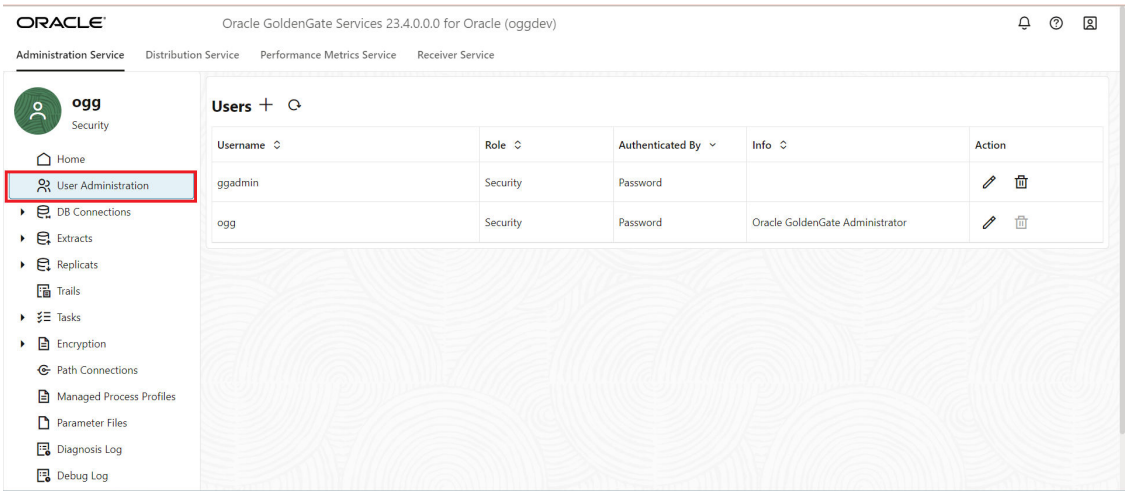
Oracle GoldenGate users that are created at different stages of deployment configuration, have different access functions for Oracle GoldenGate Microservices.

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 User Administration web page of the Service Manager. Depending on the user role, you can enable users to access Oracle GoldenGate Microservices. The following image shows the User Administration page in Service Manager with the different users.



The following image shows the User Administration page in Administration Service with the different users.



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.

Oracle GoldenGate 23ai and higher support a maximum password length of 1024 bytes.

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 ANS 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.

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 a 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 Authentication and Authorization to an External ID 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

Accountability

Oracle GoldenGate includes functional security aspects such as REST API call logging and system logging.

Auditing

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.
- 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.

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 available with Oracle GoldenGate Microservices Architecture.

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

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 a Secure Deployments

Learn about creating different types of certificates for a single deployment in a hub. In case of two deployments, learn about the options to add external certificates in cases where the distribution path needs to be established between deployments with different source and target databases.

Topics:

Create Different Types of Certificates for a Secure Deployment

Certificates are used when configuring a deployment that has a Distribution path from the Distibution and Reciver Service within a single (or same) deployment or a hub deployment.

Here's how you can create client, server certificates, and trusted chain certificates to set up a secure Oracle GoldenGate Microservices Architecture deployment.

Create a Self-Signed Trusted (Root) Certificate

You may apply your existing trusted certificate or use the `orapki` in the `OGG_HOME/bin` directory.

> **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.

Here's an example of how you can create a root certificate using `orapki`:

1. Create a directory to store your wallets and certificates. For example, `~/wallet_directory`.
2. Create an automatic login wallet. This example uses `root_ca` for the wallet name.

   ```
   orapki wallet create -wallet ~/wallet_directory/root_ca -auto_login -pwd
   welcome123
   ```

3. In the `orapki` command to create self-signed (root user) certificate, specify the`-sign_alg sha256` option.
4. In `orapki` wallet:

   ```
   orapki wallet add -wallet ~/wallet_directory/root_ca -dn "CN=RootCA" -
   addext_basic_cons -pathlen 10 -keysize 2048 -self_signed -validity 7300 -
   pwd welcome123 -sign_alg sha256
   ```

5. Export the certificate to a `.pem` file.

   ```
   orapki wallet export -wallet ~/wallet_directory/root_ca -dn "CN=RootCA" -
   cert ~/wallet_directory/rootCA_Cert.pem -pwd welcome123
   ```

The certificate creation is complete.

The following code snippet illustrates how to set up `orapki` generated wallets:

```
# 1. (Self-Signing) Root Certificate
orapki wallet create -wallet ${WORKDIR}/wallet_SRC/rootCA_SRC -auto_login -
pwd welcome123 -nologo
orapki wallet add    -wallet ${WORKDIR}/wallet_SRC/rootCA_SRC -dn
"CN=RootCA_SRC" -addext_basic_cons -pathlen 10 -keysize 2048 \
                                                                        -
self_signed -validity 1825 \
                                                                        -
pwd welcome123 -nologo
orapki wallet export -wallet ${WORKDIR}/wallet_SRC/rootCA_SRC -dn
"CN=RootCA_SRC" -cert ${WORKDIR}"/wallet_SRC/rootCA_SRC_CERT.pem" -pwd
welcome123 -nologo

# 2. Server Certificate
orapki wallet create -wallet ${WORKDIR}"/wallet_SRC/server_"${v_hostname} -
auto_login -pwd welcome123 -nologo
orapki wallet add    -wallet ${WORKDIR}"/wallet_SRC/server_"${v_hostname} -dn
"CN=${v_hostname}" \
                                                                      -
addext_basic_cons -pathlen 10 -keysize 2048 \
                                                                     -pwd
welcome123 -nologo
orapki wallet export -wallet ${WORKDIR}"/wallet_SRC/server_"${v_hostname} -dn
"CN=${v_hostname}" \
                     -request ${WORKDIR}"/wallet_SRC/server_"$
{v_hostname}"_req.pem" -pwd welcome123 -nologo
orapki cert create   -wallet ${WORKDIR}/wallet_SRC/rootCA_SRC \
                     -request ${WORKDIR}"/wallet_SRC/server_"$
{v_hostname}"_req.pem" \
                     -cert ${WORKDIR}"/wallet_SRC/server_"$
{v_hostname}"_Cert.pem" \
                     -serial_num 25 -validity 365 -pwd welcome123 -
nologo

orapki wallet add    -wallet ${WORKDIR}"/wallet_SRC/server_"${v_hostname} \
                     -trusted_cert -cert ${WORKDIR}"/wallet_SRC/
rootCA_SRC_CERT.pem" -pwd welcome123 -nologo

orapki wallet add    -wallet ${WORKDIR}"/wallet_SRC/server_"${v_hostname} \
                      -user_cert -cert ${WORKDIR}"/wallet_SRC/server_"$
{v_hostname}"_Cert.pem" -pwd welcome123 -nologo

# 3. Distribution Server Certificate
orapki wallet create -wallet ${WORKDIR}/wallet_SRC/client_SRC -auto_login -
pwd welcome123 -nologo
orapki wallet add    -wallet ${WORKDIR}/wallet_SRC/client_SRC -dn
"CN=client_SRC" -keysize 2048 -pwd welcome123 -nologo
orapki wallet export -wallet ${WORKDIR}/wallet_SRC/client_SRC -dn
"CN=client_SRC" \
                     -request ${WORKDIR}/wallet_SRC/client_SRC_req.pem   -pwd
welcome123 -nologo
orapki cert create   -wallet ${WORKDIR}/wallet_SRC/rootCA_SRC \
                     -request ${WORKDIR}/wallet_SRC/client_SRC_req.pem \
```

```
-cert ${WORKDIR}/wallet_SRC/client_SRC_Cert.pem \
-serial_num 26 -validity 365 -pwd welcome123 -nologo
```

Create Server Certificates

The following steps are an example of how you can create a sever certificate using a root certificate named `root_ca`.

1. Create a directory to store your wallets and certificates. For example, *~/wallet_directory*.
2. Create an automatic login server wallet.

   ```
   orapki wallet create -wallet ~/wallet_directory/$(hostname) -auto_login -
   pwd welcome123
   ```

   Enter the password for the server when prompted.
3. Add a Certificate Signing Request (CSR) to the server's wallet.

   ```
   orapki wallet add -wallet ~/wallet_directory/$(hostname) -dn "CN=$
   (hostname)" -keysize 2048 -pwd welcome123
   ```

   > **Note:**
   >
   > The `addext_basic_cons -pathlen 10` option is important as it is used to apply the later certificate into another secure store, when setting certificates for two different deployments. See Create a RootCA External Certificate in the Target Deployment. For more information, refer to Connecting Two Deployments Using External RootCA Certificate.

4. Export the CSR to a `.pem` file.

   ```
   orapki wallet export -wallet ~/wallet_directory/$(hostname) -dn "CN=$
   (hostname)" -request ~/wallet_directory/servername_req.pem -pwd welcome123
   ```

5. Using the CSR, create a signed server certificate and sign it using the root certificate. Assign a unique serial number to each certificate.

   ```
   orapki cert create -wallet ~/wallet_directory/root_ca -request ~/
   wallet_directory/servername_req.pem -cert ~/wallet_directory/
   servername_Cert.pem -serial_num 20 -validity 375  -sign_alg sha256
   ```

6. Add the root certificate into the server's wallet as a trusted certificate.

   ```
   orapki wallet add -wallet ~/wallet_directory/$(hostname) -trusted_cert -
   cert ~/wallet_directory/rootCA_Cert.pem -pwd welcome123
   ```

7. Add the server certificate as a user certificate into the server's wallet.

   ```
   orapki wallet add -wallet ~/wallet_directory/$(hostname) -user_cert -cert
   ~/wallet_directory/servername_Cert.pem  -pwd welcome123
   ```

The wallet creation is complete.

Create a Client Certificate

The following steps are an example of how you can create a Distribution Service user certificate:

1. Create a directory to store your wallets and certificates. For example, `~/wallet_directory`.
2. Create an automatic login client wallet. This example uses `dist_client` for the wallet name.

   ```
   orapki wallet create -wallet ~/wallet_directory/dist_client -auto_login -
   pwd welcome123
   ```

3. Add a CSR to the wallet.

   ```
   orapki wallet add -wallet ~/wallet_directory/dist_client -dn
   "CN=dist_client" -keysize 2048 -pwd welcome123
   ```

4. Export the CSR to a `.pem` file.

   ```
   orapki wallet export -wallet ~/wallet_directory/dist_client -dn
   "CN=dist_client" -request ~/wallet_directory/dist_client_req.pem -pwd
   welcome123
   ```

5. Using CSR, create a signed client certificate and sign it using the root certificate. Assign a unique serial number to each certificate.

   ```
   orapki cert create -wallet ~/wallet_directory/root_ca -request ~/
   wallet_directory/dist_client_req.pem -cert ~/wallet_directory/
   dist_client_Cert.pem -serial_num 30 -validity 375 -pwd welcome123
   ```

6. Add the root certificate as a trusted certificate into the client's wallet.

   ```
   orapki wallet add -wallet ~/wallet_directory/dist_client -trusted_cert -
   cert ~/wallet_directory/rootCA_Cert.pem -pwd welcome123
   ```

7. Add the client certificate as a user certificate into the client's wallet.

   ```
   orapki wallet add -wallet ~/wallet_directory/dist_client -user_cert -cert
   ~/wallet_directory/dist_client_Cert.pem -pwd welcome123
   ```

The wallet creation is complete.

Set Up Trusted Certificates

There are two types of TLS connections. To use TLS, there are certain requirements for the certificate trust chain.

The `wss` communication protocol is used in the Distribution Service for the Distribution Path to meet the needs of secure communication using TLS in Oracle GoldenGate Microservices Architecture.

> **Note:**
>
> Adding a non-CA self-signed certificate as a trusted certificate using Service Manager's Certificate Management web interface's CA Cert section is not supported and will result in an error.

Setting up the server's CA certificate as a Trusted Certificate for External Identity Provider

To work with an external Identity Provider (IDP) such as IDCS, you need to upload the IDP server's (IDCS) CA certificate as a trusted certificate.

See Manage Certificates for Deployments.

Distribution Service and Receiver Service

Both the Distribution Service and Receiver Service need certificates. The Distribution Service uses the certificate in the client wallet location under outbound section. The location of that wallet can be found in the `deploymentConfiguration.dat` file under *`deployment_home`*`/etc/conf`.

The certificates in both wallets need to be trusted by each other, so either both need to have commercial certificates issued by Oracle GoldenGate Microservices Architecture, or they have to trust each other using self-signed certificates.

For self-signed certificates, you can choose from one of the following:

- Have both certificates signed by the same root certificate. (`rootCA`)
- The other side's certificate is added to the local wallet as a trusted certificate

For the Receiver Service, the certificate is in the wallet for the local wallet location, which is also in the `deploymentConfiguration.dat` file.

On the Distribution Service, if the hostname used in the Receiver Service's certificate can't be routed correctly, `/etc/hosts` file should be updated with the correct IP address for that host. The Distribution Service will use this IP address to communicate with the Receiver Service once it accepts the certificate from the Receiver Service.

Using the Reverse Proxy (NGINX) with the Distribution Service and Receiver Service

You only need to add the Nginx certificate to the Distribution Service's client wallet as a trusted certificate. Usually the certificate used by NGINX is self-signed.

The host name in the Nginx certificate should also be routable. If not, on the Distribution Service, `/etc/hosts` file needs to be updated to reflect the correct IP address for that host name.The Distribution Service will use the host name in the certificate to communicate to the target. If the NGINX certificate doesn't have a valid host name in it, but has a Subject Alternative Name record, then the host name is the DNS name there.

See Configure Reverse Proxy with NGINX to Access Oracle GoldenGate Microservices

Create a RootCA External Certificate in the Target Deployment

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

Here's a sample of the client certificate configuration (`client_src_to_trg.cfg`) file:

```
[ req ]
default_bits = 4096
default_md = sha512
prompt = no
encrypt_key = no
distinguished_name = req_distinguished_name
[ req_distinguished_name ]
commonName = "client_src_to_trg"
[ my_extensions ]
```

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

```
# client certificate
openssl req -new -newkey rsa:2048 -nodes -keyout client.key -out client.csr -
config client.cfg
openssl x509 -req -days 73000 -in client.csr -CA rootCA_extern.cert -CAkey
rootCA_extern.key -CAcreateserial -out client.cert
```

Here is a sample `rootCA_extern.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_extern"
#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:

```
# rootCA certificate
openssl req -x509 -newkey rsa:4096 -keyout rootCA_extern.key -out
rootCA_extern.cert -days 73000 -nodes -config rootCA_extern.cfg
```

Configure Reverse Proxy with NGINX to Access Oracle GoldenGate Microservices

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

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.

When sending trail files from Oracle GoldenGate Classic to Microservices environment that is configured with a reverse proxy, use a pump Extract from Oracle GoldenGate Classic with `SOCKSPROXY` option. When sending trail files from Oracle GoldenGate Microservices to Classic Architecture use the `ogg` protocol in the Distribution Service configuration.

See Connect Oracle GoldenGate Classic Architecture to Microservices Architecture and Connect Oracle GoldenGate Microservices Architecture to Classic Architecture in *Administering Oracle GoldenGate*.

You can configure Oracle GoldenGate Microservices Architecture to use a reverse proxy. Oracle GoldenGate MA includes a script called `ReverseProxySettings` that generates configuration file for only the NGINX reverse proxy server.

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.

You can use these options by running the `ReverseProxySettings` utility. Here are the options available with this utility:

`-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.

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

Topics:

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. 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. Replace the existing NGINX configuration with the configuration that was generated using the `ReverseProxySetting` utility for your MA deployment:

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

However, this NGINX configuration isn't complete without the `events` section, and enclosing the `map` and `server` sections in `http`.

Optionally, you can use the default `nginx.conf` file and add the generated `ogg.conf` by adding an `include` statement similar to this:

```
include /etc/nginx/conf.d/ogg.conf;
```

In this case, you must comment out the other `servers` section.

4. Generate a self-signed certificate for NGINX:

```
sudo sh /etc/ssl/certs/make-dummy-cert /etc/nginx/ogg.pem
```

For distribution paths to go through the reverse proxy, you need to use a valid certificate. It's better to specify the same certificate that the deployment is using to process incoming requests, otherwise, starting the path will fail with the next error in Distribution Service:

```
2019-03-26T11:26:00.324-0700 ERROR| ERROR OGG-10351  Oracle GoldenGate
Distribution
 Service for Oracle:  Generic error -1 noticed. Error description -
Certificate validation
 error: Unacceptable certificate from test00abc: application verification
failure. (A4)
```

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. Reload NGINX with the new configuration:

```
sudo nginx -s reload
```

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.

Encrypting Trail Files

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

Topics:

Generate Master Keys and Encryption Key

You can generate the master key and encryption keys using the **Key Management** tab in the **Configuration** page of the Administration Service.

Using Master Keys

If you want to encrypt your data, then create a Master Key by clicking the + sign in the Master Key section. The master key is generated automatically.

You can change the status of the key to Available or Unavailable, by clicking the edit icon in the Master Key table. You can also delete the Master Key from the table by clicking the delete icon.

Using the Encryption Keys

To use this method of data encryption, you configure Oracle GoldenGate to generate an encryption key and store the key in a local `ENCKEYS` file. The `ENCKEYS` file must be secured through the normal method of assigning file permissions in the operating system. This procedure generates an AES encryption key and provides instructions for storing it in the `ENCKEYS` file.

To generate the ENCKEYS files, click the + sign in the Encryption Keys section. The Encryption Key is generated.

Key Management Service (KMS)

Oracle GoldenGate supports Oracle Key Vault (OKV) and Oracle Cloud Infrastructure Key Management Service (OCI KMS) methods to manage encryption keys.

Oracle GoldenGate Microservices Architecture supports KMS to provide scalability in managing encryption keys and credentials along with security such that the key isn't stored or managed by Oracle GoldenGate.

Oracle GoldenGate uses the encapsulation approach to encrypt trail files. It generates a data encryption key (DEK) for each trail file, known as local key. 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. This process is called encapsulation encryption.

In Oracle GoldenGate, a KMS can be used to manage cryptographic keys within an enterprise.

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 ANS 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.

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)

Configure an Encryption Profile

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

To set up the encryption profile, click **Profile** from the navigation pane and then select the **Key Management System** (KMS) tab.

1. By default, the **Local Wallet** profile is created. If you select the **Local Wallet** encryption profile, you'll see its options, which you can edit using the pen icon.

Options	Description
Description	A description of the local wallet.
Default Profile	This option is enabled by default. You can select to disable it.
Encryption Profile Type	This option cannot be changed for the local wallet.
Masterkey Name	`OGG_DEFAULT_MASTERKEY` default master key for the local wallet. You cannot edit this value.
Masterkey Version	This is the master key version number. The value is set to **LATEST** and cannot be changed.

2. Click the + sign next to **Profile** to create an encryption profile by specifying the following details:

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).

3. 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.

4. For configuring the encryption profile for OCI KMS, see Using OCI KMS Trail File Encryption in Oracle GoldenGate.

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.

Topics:

Oracle Key Vault Capabilities

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 OKV on Oracle GoldenGate

Learn the prerequisites for setting up OKV with Oracle GoldenGate.

The following steps belong to the OKV configuration on the machine where the Oracle GoldenGate instance is running:

1. Download the `okvrestservices.jar` from the OKV server, where Oracle GoldenGate is deployed as the same system user as the deployment.
2. 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
   ```

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

   ```
   OS> java -jar okvrestservices.jar kmip
           --config /u01/app/oracle/OKV/conf/okvclient.ora
           --service create_key
           --algorithm AES
           --length 256
           --mask
   "ENCRYPT,DECRYPT,TRANSLATE_ENCRYPT,TRANSLATE_DECRYPT,TRANSLATE_WRAP,TRANSLA
   TE_UNWRAP"
           --wallet OKV_WALLET76876ABA-B06D-4F35-BF7C-D9306D29764B
   ```

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

   ```
   OS>java -jar okvrestservices.jar kmip
               --config ./conf/okvclient.ora --service reg_key -
   ENCRYPT,DECRYPT,TRANSLATE_ENCRYPT,TRANSLATE_DECRYPT,TRANSLATE_WRAP,TRANSLAT
   E_UNWRAP
               --wallet OGG_WALLET
               --object /u01/key.txt64B3AAD0-BE77-1821-E053-0100007FD178
   ```

4. 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 Administration Guide* for details about the configuration within the OKV server.

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

   ```
   OS> java -jar okvrestservices.jar kmip
               --config /u01/app/oracle/OKV/conf/okvclient.ora
               --service activate
               --uid 76876ABA-B06D-4F35-BF7C-D9306D29764B
   INFO: Success
   ```

6. 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> java -jar okvrestservices.jar kmip
               --config /u01/app/oracle/OKV/conf/okvclient.ora
               --service add_custom_attr
   ```

```
            --uid 76876ABA-B06D-4F35-BF7C-D9306D29764B
            --attribute x-OGG-KeyName
            --type TEXT
            --value OGG_Masterkey
INFO: Success


OS> java -jar okvrestservices.jar kmip
            --config /u01/app/oracle/OKV/conf/okvclient.ora
            --service add_custom_attr
            --uid 76876ABA-B06D-4F35-BF7C-D9306D29764B
            --attribute x-OGG-KeyVersion
            --type TEXT
            --value 1
INFO: Success
```

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

```
OS>okvutil list -v 4
okvutil version 18.2.0.0.0
Endpoint type: Oracle (non-database)
Configuration file: /u01/app/oracle/OKV/conf/okvclient.ora
Server: 10.245.64.45:5696 10.245.64.46:5696
Standby Servers:Read Servers: 10.245.64.48:5696
Auto-login wallet found, no password needed
Trying to connect to 10.245.64.45:5696 ...
Connected to 10.245.64.45:5696.
Unique ID Type Identifier
72B673E8-840B-4AD6-8400-CB77B68D74B5 Template Default template for OGG_EP
76876ABA-B06D-4F35-BF7C-D9306D29764B Symmetric Key -
```

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

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
Administation 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: `ADD ENCRYPTIONPROFILE`, `ALTER ENCRYPTIONPROFILE`, `DELETE ENCRYPTIONPROFILE`, `INFO ENCRYPTIONPROFILE`. In addition, the `ADD` or `ALTER` the Extract, DISTPATH, or Replicat commands have been modified to include the parameter `ENCRYPTIONPROFILE` *`encryption-profile-name`*. To know more, see Admin Client Command Line Interface Commands in *Command Line Interface Reference for Oracle GoldenGate*.

Client Behavior Against Different Key States for Oracle Key Vault

This topic describes the relative behavior of the of the reader or writer client processes depending on the different encryption key states.

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.

Key State	Trail Writer (encryption)	Trail Reader (decryption)
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.

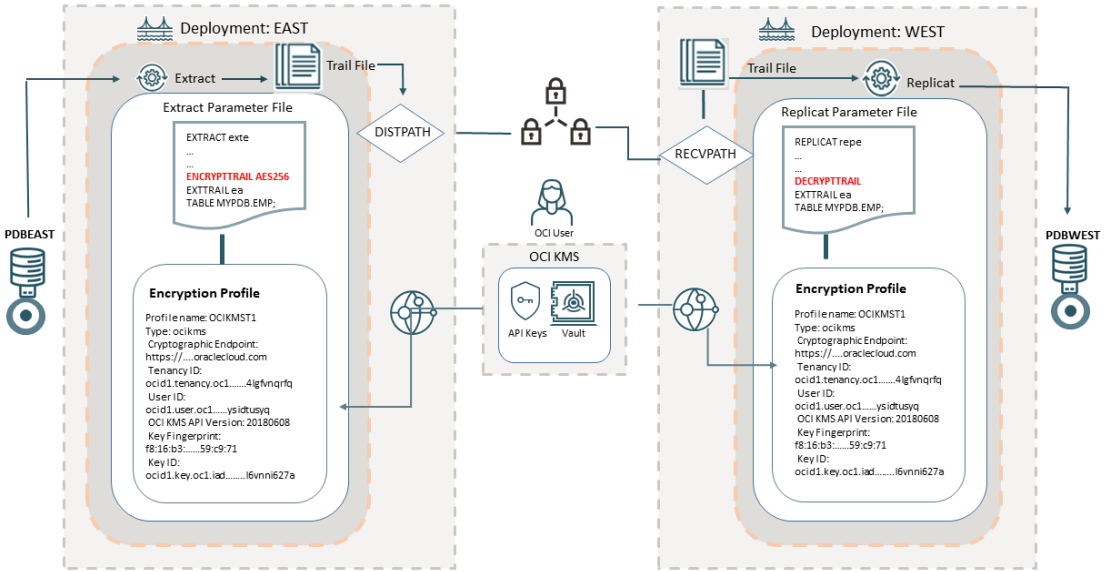
Using OCI KMS Trail File Encryption in Oracle GoldenGate

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.

Topics:

Oracle GoldenGate with OCI KMS Workflow

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



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.

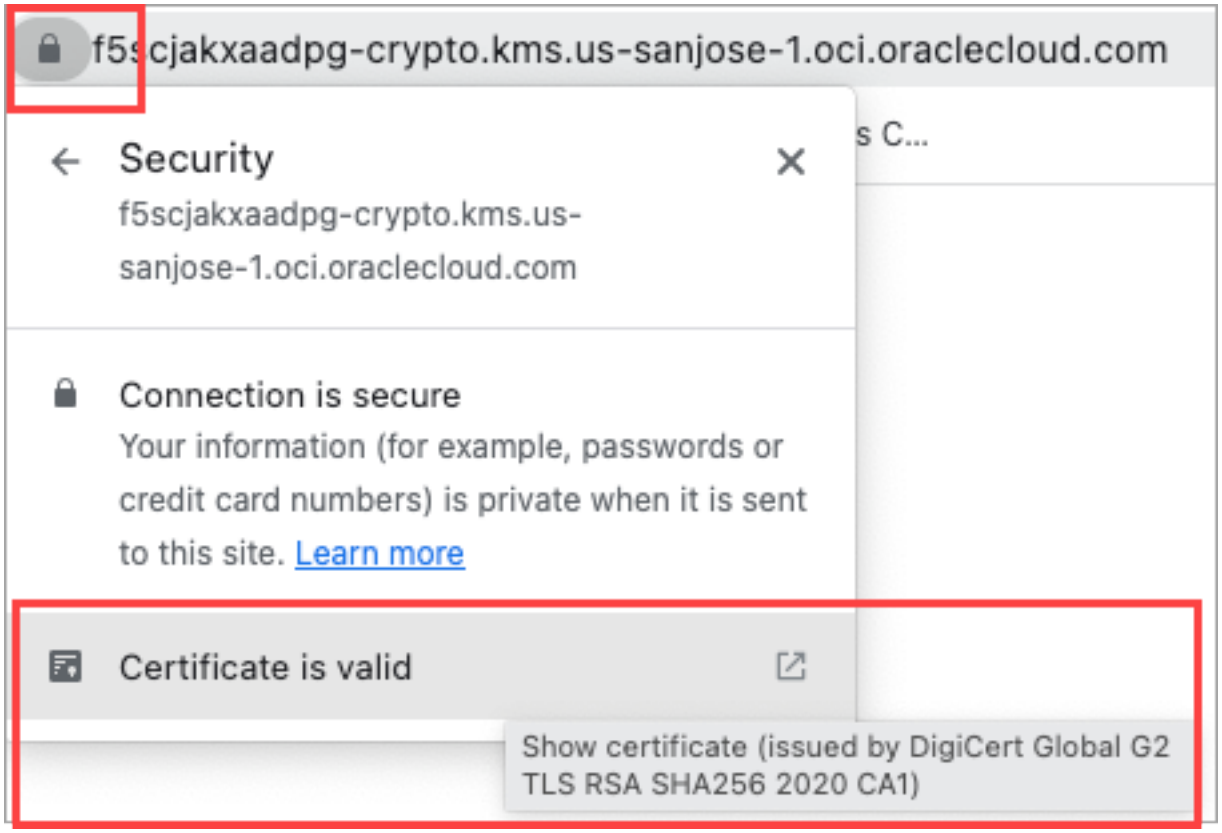
Topics:

Download the CA Certificate using the Cryptographic Endpoint

To perform the steps in this topic, you need to have a Vault in your OCI teanancy 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 teanancy. 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.

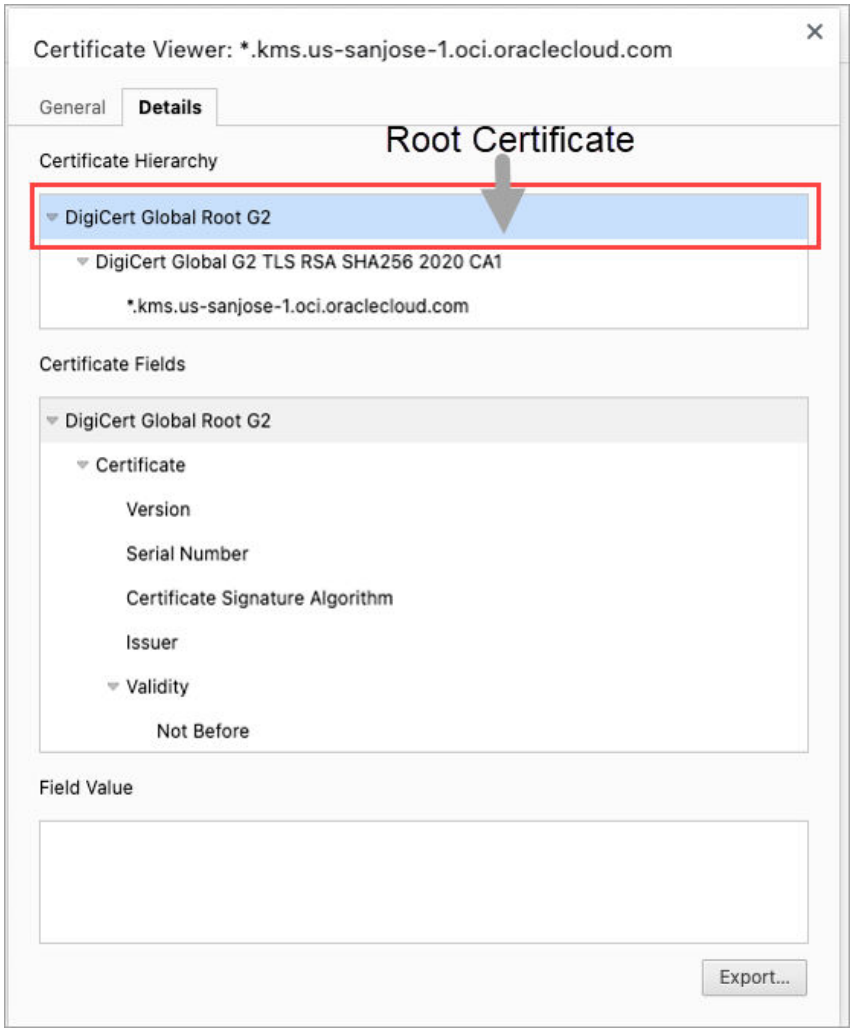
If you have an existing Vault in your OCI tenancy, then follow the steps provided in this topic, to download the CA certificate using the cryptographic endpoint.

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.

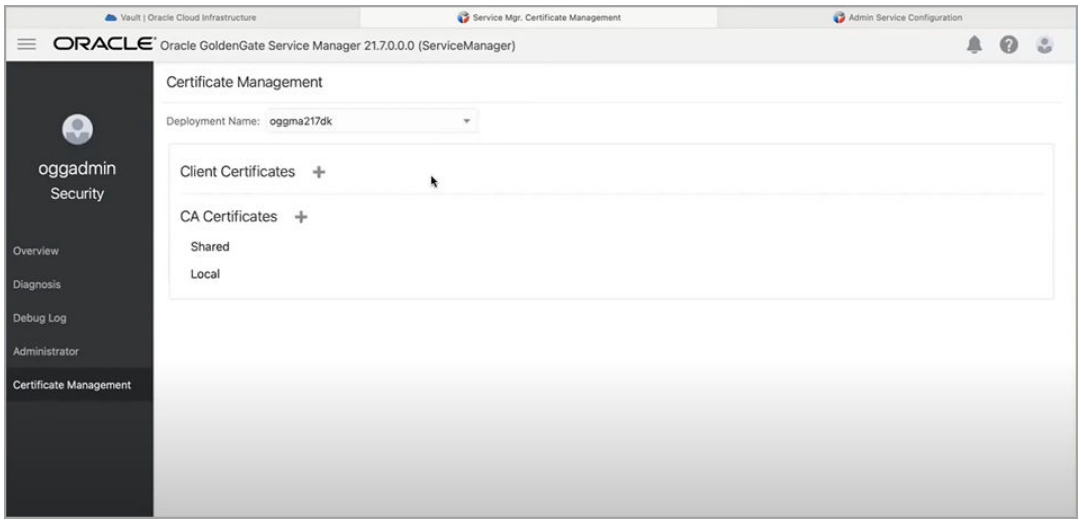
Add the Digital CA Certificate as a Trusted CA Certificate in Oracle GoldenGate

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 Set Up Trusted Certificates.

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

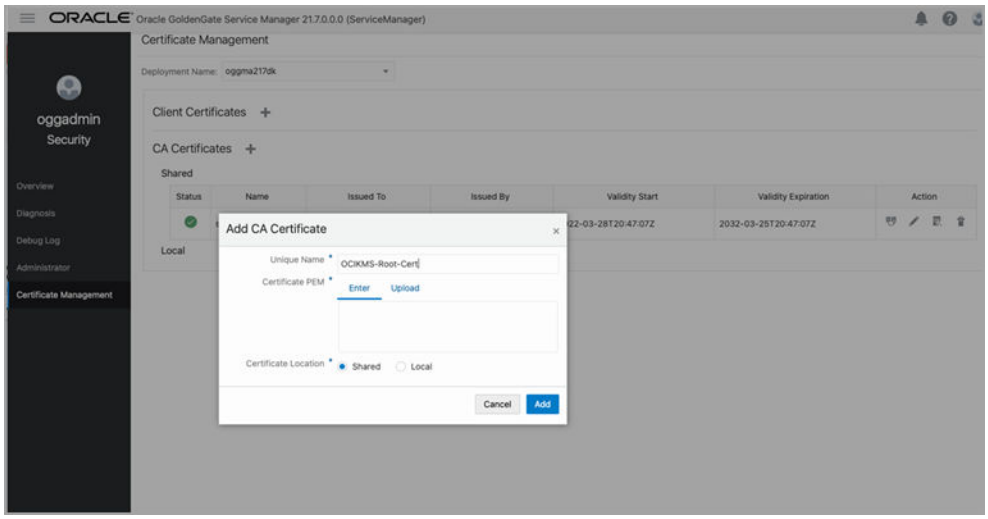
From the Oracle GoldenGate Service Manager, perform the following steps to add the digital CA certificate as a Trusted CA certificate for the source deployment:

1. Log in to Oracle GoldenGate Service Manager.
2. From the left-navigation pane, select the Certificate Management option.



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.



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:

Topics:

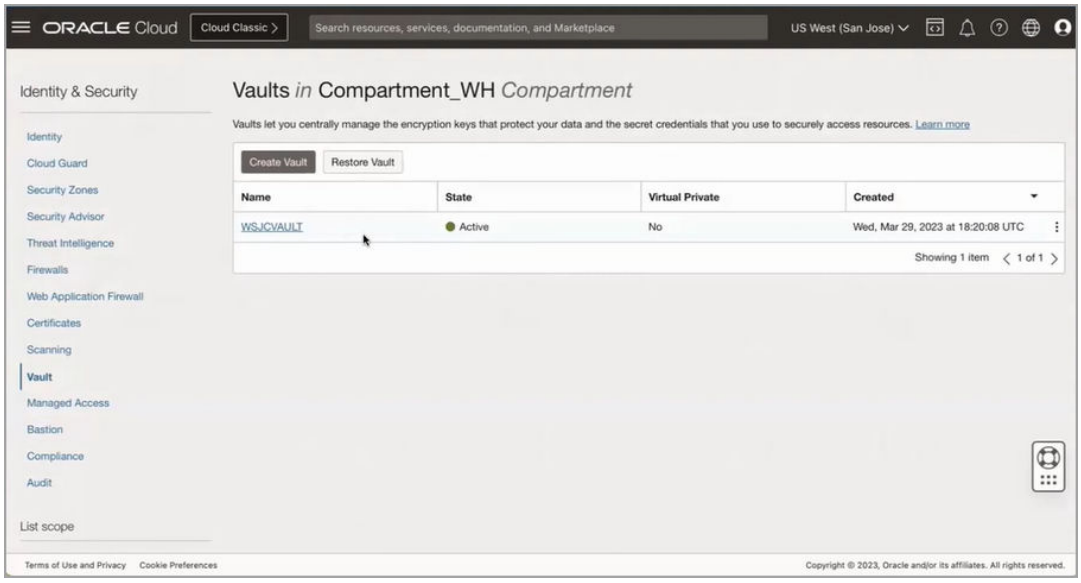
Create or Access the OCI Vault

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

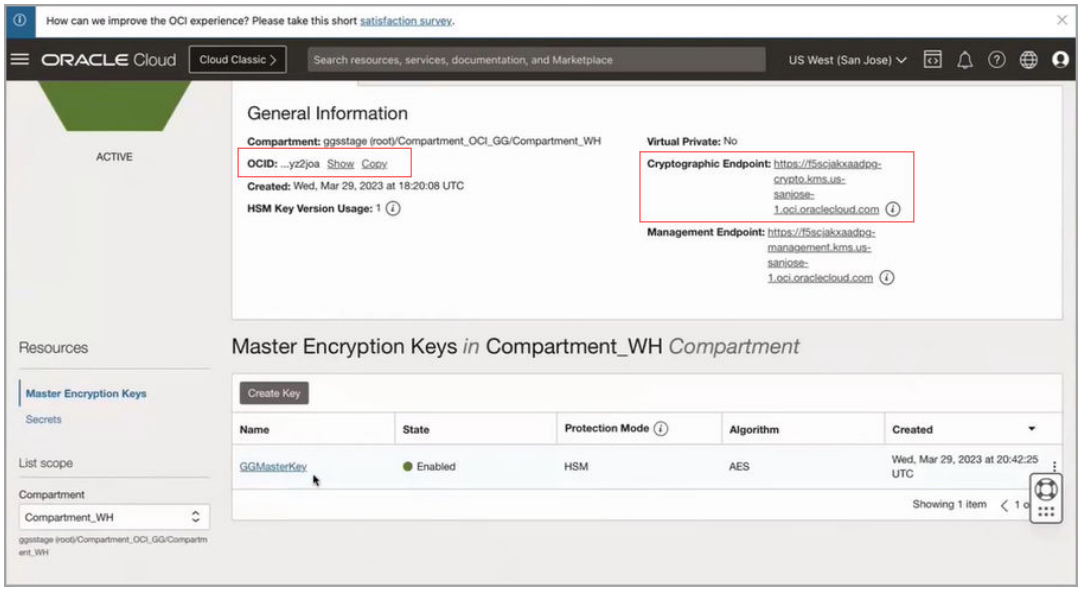
- **Cryptographic Endpoint:** This is the link from where you need to obtain the trusted certificate. Copy this link for use in the later steps.
- **OCID**: This is the unique ID for your OCI environment. It will be required while setting up the encryption profile in Oracle GoldenGate.

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. Copy this information to a notepad for reference.

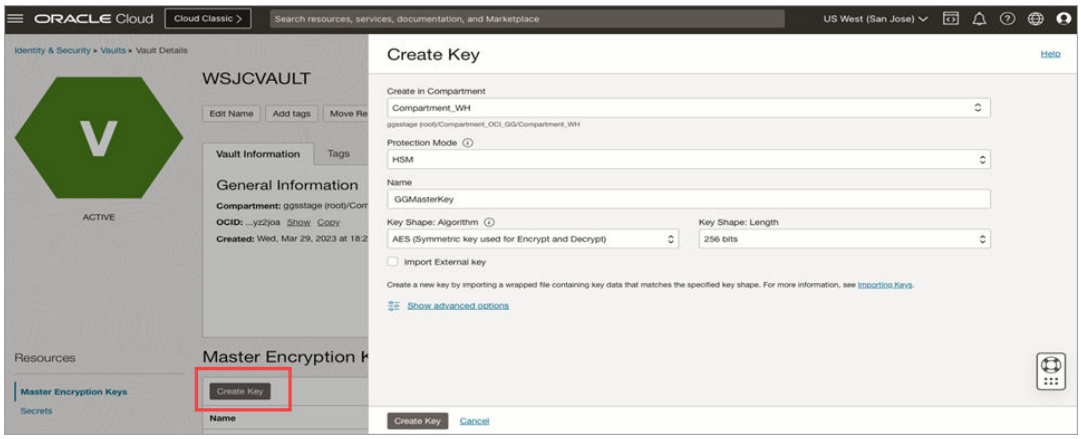
Generate the Master Key and Download the API Private Key

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. Click **Create Key** to display the Create Key page.
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.

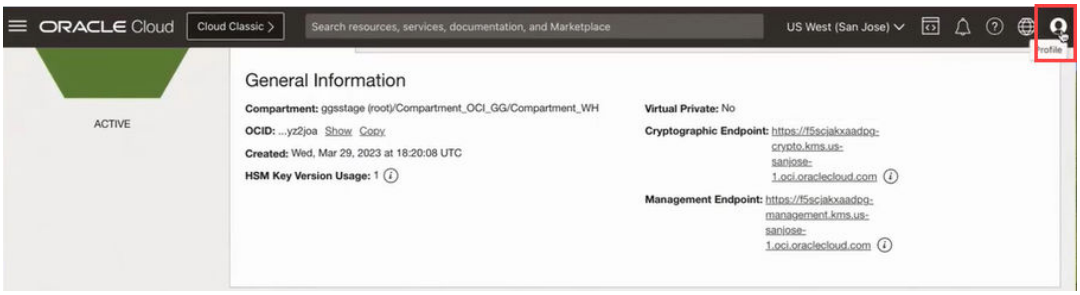


4. Click **Create Key**. This generates the master key that would be used by Oracle GoldenGate.

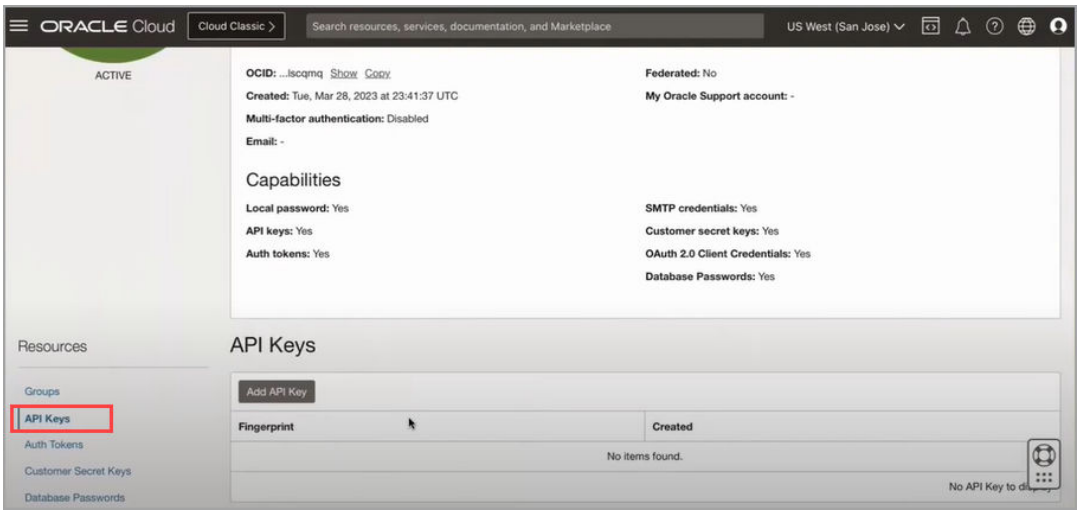
Generate API Key

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.

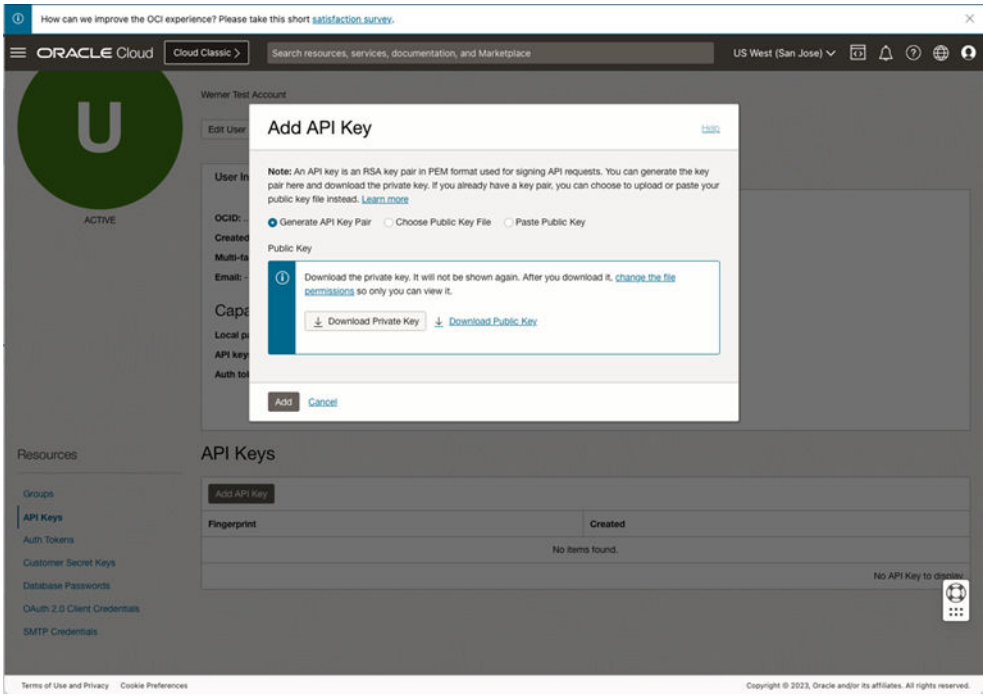


2. Click the **API Key** option from the Resource section of the left panel to open the API Keys section.



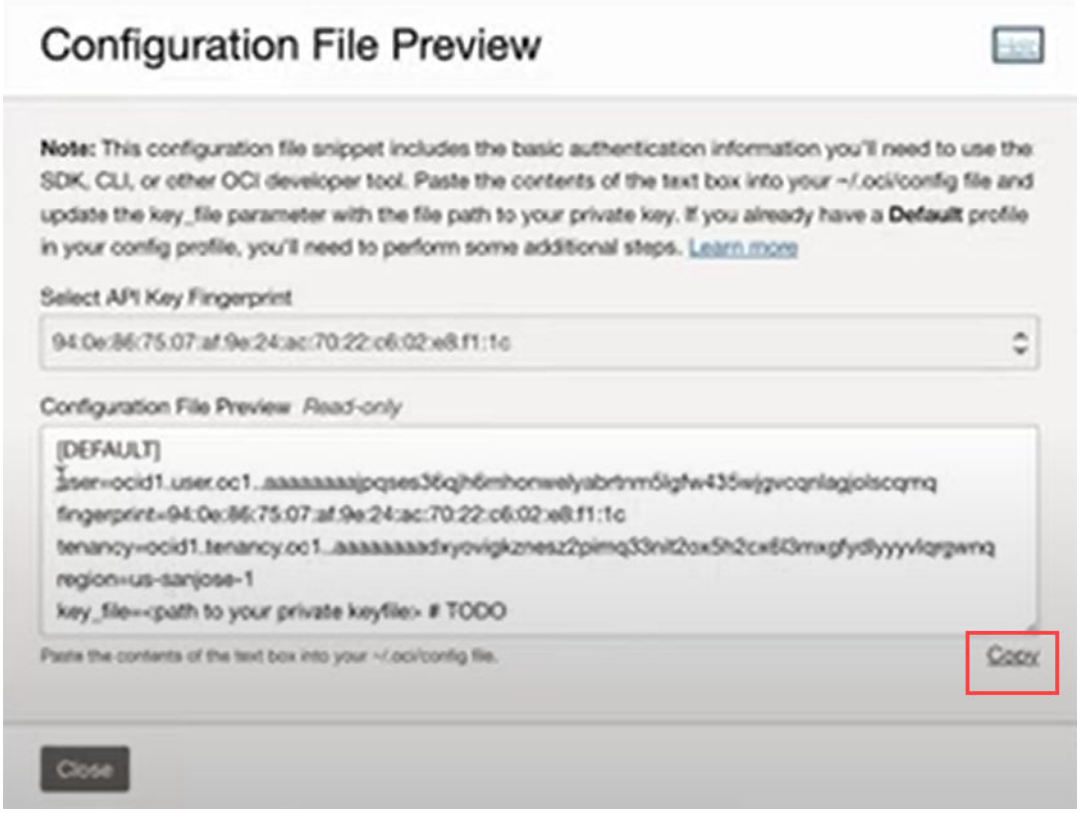
3. Click **Add API Key** to open the Add API Key dialog box.
4. Select the **Generate the API key Pair** option to create a key pair for the OCI user to connect with OCI KMS. You also have the option to upload an existing public key file using the **Choose Public Key File** option or paste the value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

the **Paste Public Key** option.

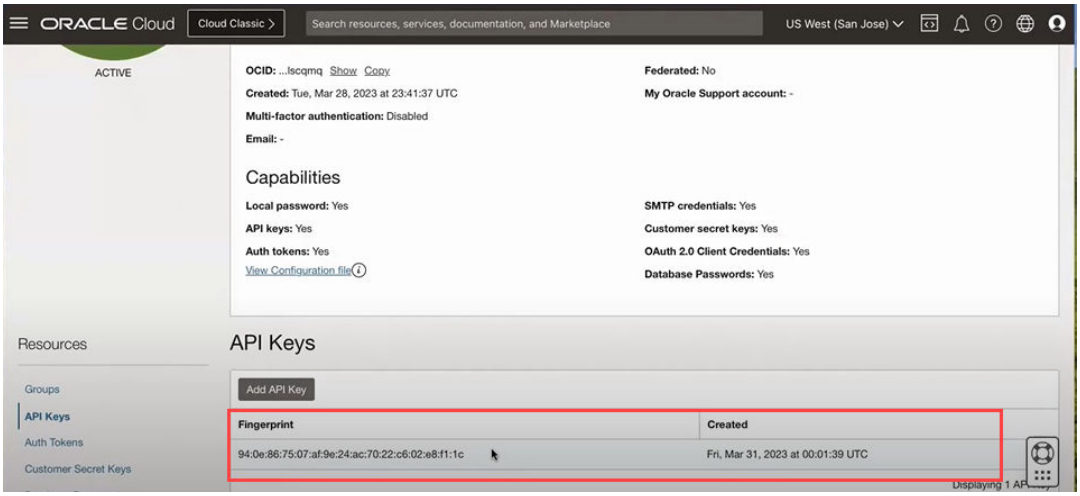


5. Click **Download Private Key** and keep it in a known location. You can rename the file to a user-friendly name such as `API_private_key.pem`.
6. Click **Add**. This displays the **Configuration File Preview** dialog box, which contains all information associated with the API key such as the fingerprint,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

> **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.



7. 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.

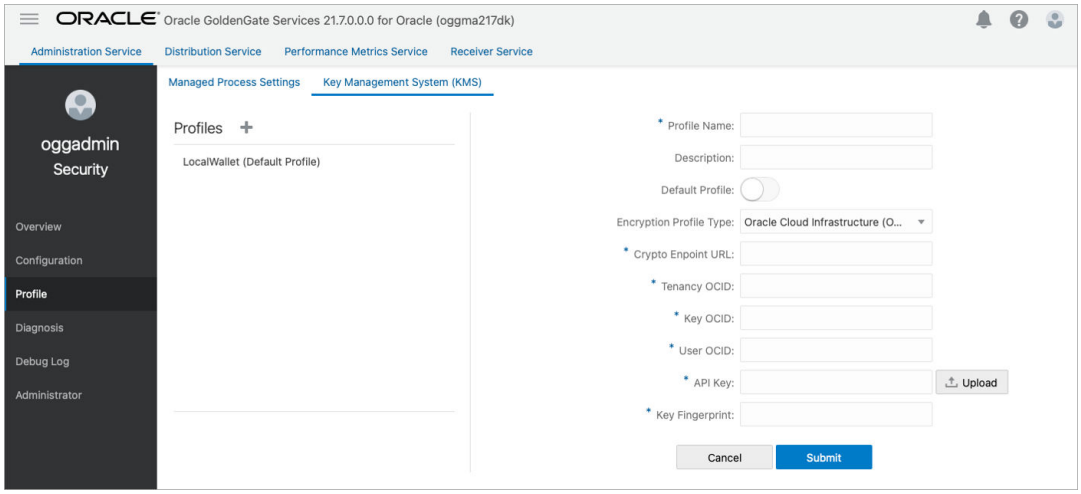
Use the following steps to apply OCI KMS-based trail encryption from Oracle GoldenGate Microservices Architecture:

Topics:

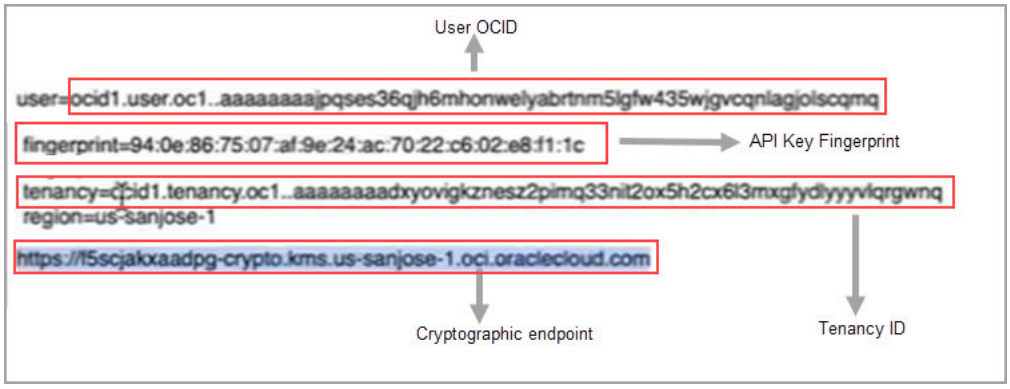
Create Encryption Profile in Oracle GoldenGate Processes

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

1. Log in to the Administration Service and select the **Profile** option from the left-navigation pane.
2. Click the **Key Management System (KMS)** tab.



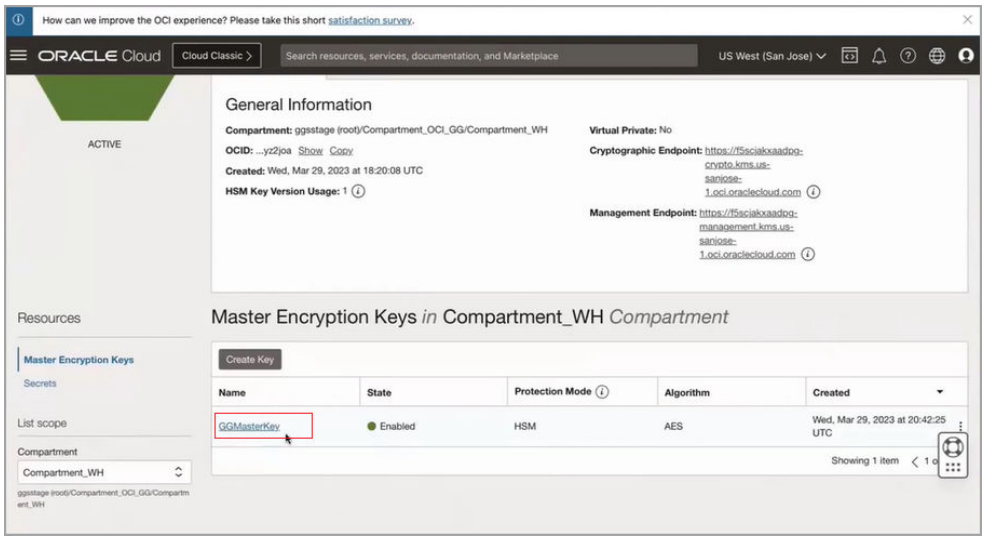
3. Open the notepad file where you saved the details for the OCI KMS API key and crypto endpoint details. See step 8 for reference from the Configure OCI KMS to Connect with Oracle GoldenGate. The following image displays the values obtained from API Configuration File Preview dialog box:



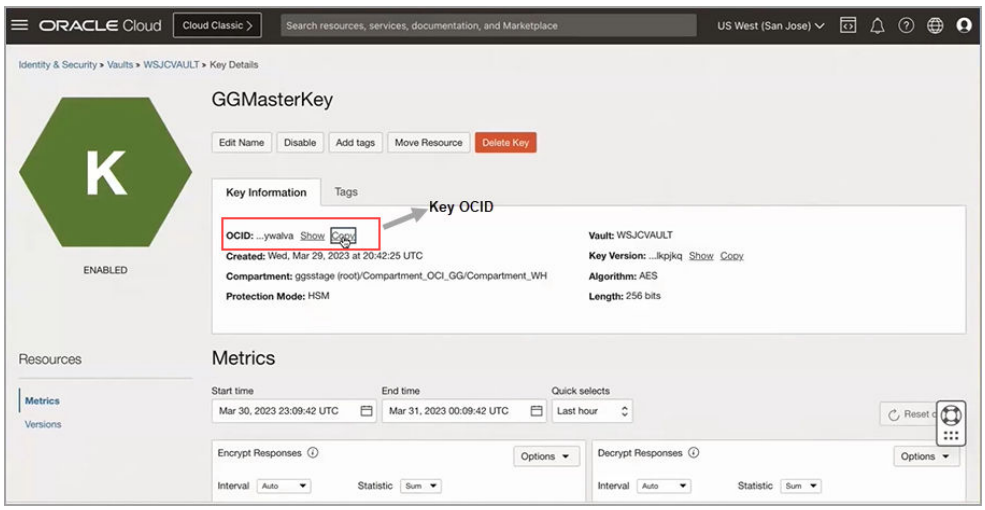
The information would include the following values:

- 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 API key to open the key details page where the OCID for the API key is provided.

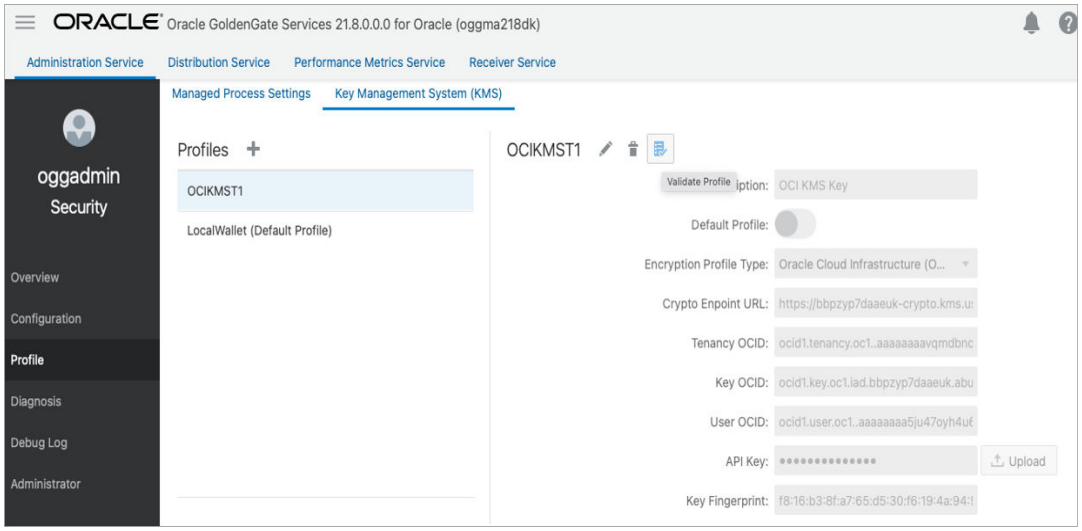


  b. Copy the API private key OCID from the Key Details page.

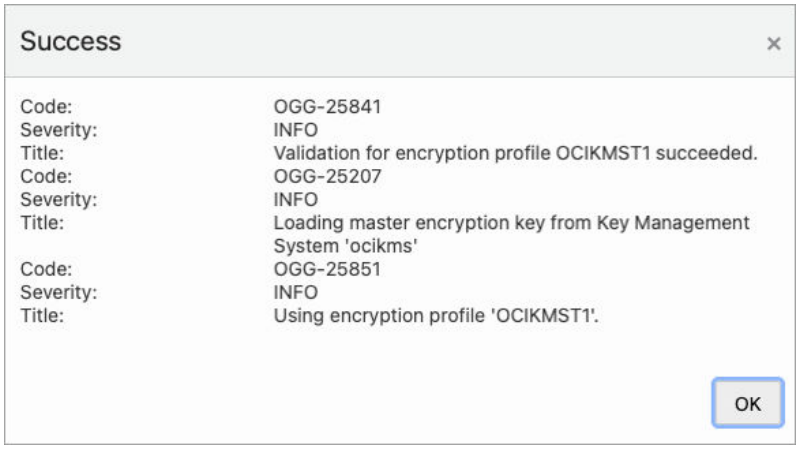


- User OCID: Obtain this value from the API configuration details.
- API Private Key: Upload the API Private Key from the location where you saved it while performing tasks in Configure OCI KMS to Connect with Oracle GoldenGate.
- API Key Fingerprint: Obtain this value from the API configuration details.

4. Validate the encryption profile.



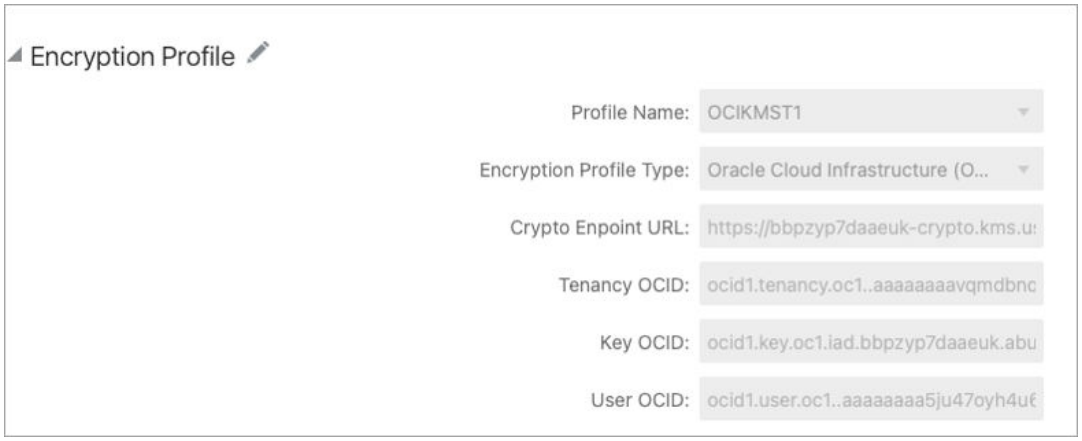
You will see a message box similar to the following confirming that the validation of the encryption profile was successful.



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 Overview page, click **Add Extract**.
2. After providing other details for the Extract, scroll down and expand the **Encryption Profile** section and select **OCIKMS** profile, such as **OCIKMST1**.



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

```
EXTRACT ktst
USERIDALIAS ggwest DOMAIN OracleGoldenGate
ENCRYPTTRAIL AES256
EXTTRAIL tt
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 OCIKMSTS1). Click **Next**.
8. In the Replicat parameter file, include the `DECRYPTTRAIL` option. The Replicat parameter file looks similar to the following:

```
REPLICAT renct
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.

Topics:

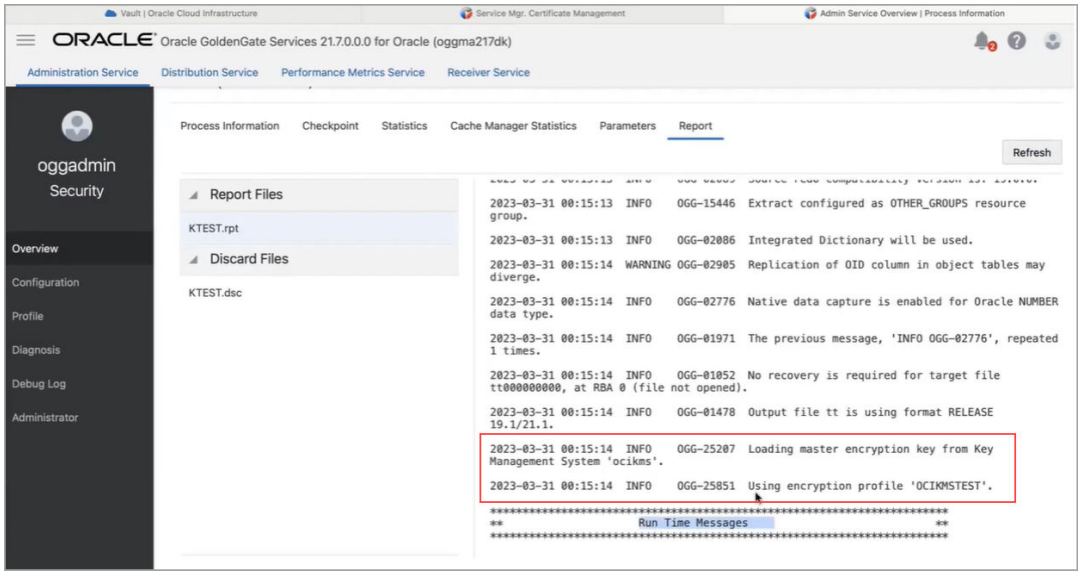
Test Trail File Encryption in the Source Deployment

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**, **Details**, **Report** tab to view the Extract report file.

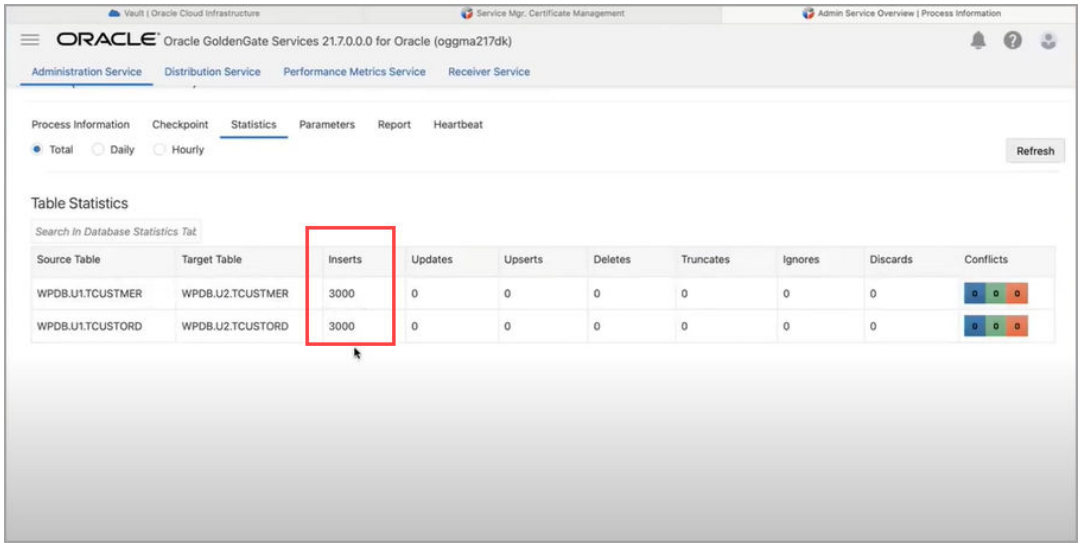


2. For troubleshooting purposes, you can check if the trail file is encrypted at source, using Logdump commands. See the Scanning a Trail File to Check for Trail File Encryption from the *Logdump Reference for Oracle GoldenGate*.

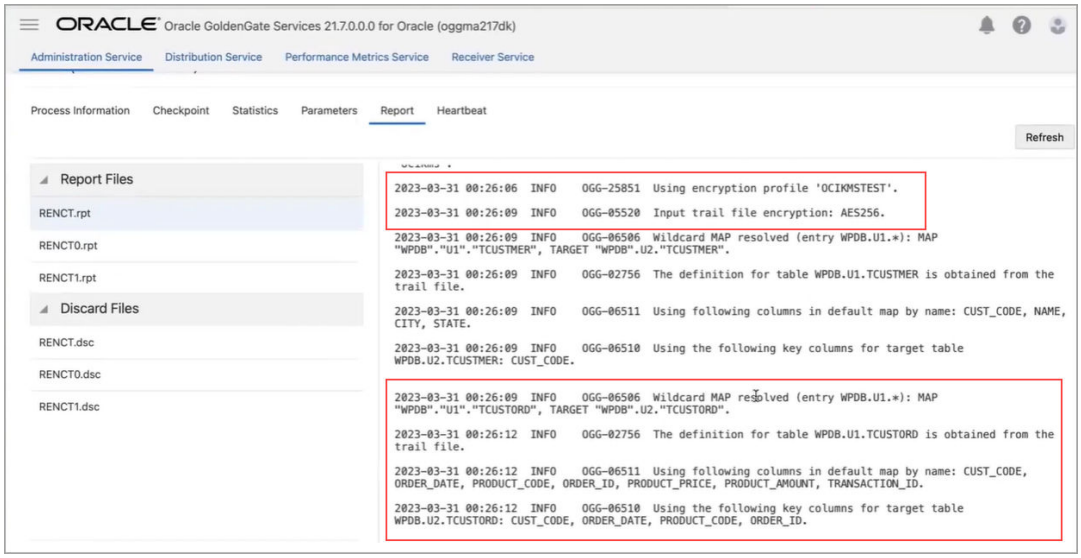
Test the Trail File Decryption on the Target Deployment

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 Overview page, click **Replicat**, **Details**, **Statistics**.
2. On the Statistics page, the applied transactions are displayed as shown in the following figure:



3. Check the Replicat report file 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.

Starting with Oracle GoldenGate 23c, maximum password length has been increased to 1024 bytes.

It is the alias, not the actual user ID or password, that is specified in a command or parameter file, and no user input of an encryption key is required. The credential store is implemented as an autologin wallet within the Oracle Credential Store Framework (CSF).

Another benefit of using a credential store is that multiple installations of Oracle GoldenGate can use the same one, 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`.

Topics:

Credential Store Tasks

1. (Optional) To store the credential store in a location other than the `dircrd` subdirectory of the Oracle GoldenGate installation directory, specify the desired location with the `CREDENTIALSTORELOCATION` parameter in the `GLOBALS` file.
2. From the Oracle GoldenGate installation directory, start the command line.
3. After using the `CONNECT` command to login to the deployment (when using the Admin Client), isssue the following commands to perform various tasks with the credential store.

Command	Description
`ADD CREDENTIALSTORE`	Adds a database credential store.
`ALTER CREDENTIALSTORE`	Adds each set of credentials to the credential store.
`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.

Specifying the Alias in a Parameter File or Command

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

Table 10-2 Specifying Credential Aliases in Parameters and Commands

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

Encrypting and Storing 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  krbtst/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 MININUSERIDLIAS ggadmin_mining
DOMAIN OracleGoldenGate
```

11
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.

For Oracle GoldenGate Extract for MySQL to process DDL replication, it is a requirement that the table in process exists in the database. Extract will not be able to process the DDL and abend, if that table is dropped from the database or table does not exist in the database. A warning message is issued indicating that the table is not found in the database. In such cases, you need to remove that table from the Extract list and then restart the Extract process.The DDL operation `CREATE TABLE AS SELECT` is not supported. Oracle GoldenGate Extract for MySQL abends with the following error when it encounters this DDL.

```
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.
```

> **Note:**
>
> See `TRANLOGOPTIONS WARNCREATEASSELECT` in the *Parameters and Functions Reference for Oracle GoldenGate* for details.

MySQL: Prerequisites for Transaction Log Based DDL Configuration

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` and `binlog_row_metadata` variables 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.

Plug-in Based DDL Configuration Prerequisites and Considerations

This is an older approach to performing DDL Replication. The prerequisites for configuring DDL replication are as follows:

- DDL replication is supported for MySQL 5.7.
- Remote capture for MySQL 5.7 doesn't support DDL replication.
- Oracle GoldenGate DDL replication uses two plug-ins as a shared library, `ddl_rewriter` and `ddl_metadata`, which must be installed on your MySQL server before Oracle GoldenGate replication starts.
- The standalone application, Oracle GoldenGate `metadata_server`, must be running to capture the DDL metadata.
- The `history` table under the new `oggddl` database (`oggddl.history`). This metadata history table is used to store and retrieve the DDL metadata history. The history table records must be ignored from being logged into the binary log so you must specify `binlog-ignore-db=oggddl` in the `my.cnf` file.
- You should not manually drop the `oggddl` database or the `history` table because all DDL statements that run after this event will be lost.
- You should not stop the `metadata_server` during DDL capture as all the DDL statements that run after this event will be lost.
- You should not manually remove the `ddl_rewriter` and the `ddl_metadata` plugins during DDL capture because all DDL statements that run after this event will be lost.

- DDL executed within the stored procedure is *not* supported. For example , a DDL executed as in the following is *not* supported.

```
CREATE PROCEDURE atssrc.generate_data()
BEGIN
DECLARE i INT DEFAULT 0;
WHILE i < 800 DO
SET i = i + 1;
IF (i = 100) then
alter table atssrc.`ddl6` add col2 DATE after id;
ELSEIF (i = 200) then
alter table atssrc.`ddl6` add col3 DATETIME after datetime;
ELSEIF (i = 300) then
alter table atssrc.`ddl6` add `col4` timestamp NULL DEFAULT NULL after
channel;
ELSEIF (i = 400) then
alter table atssrc.`ddl6` add col5 YEAR after value;
END IF;
END WHILE;
END$$
DELIMITER ;
call atssrc.generate_data();
```

- By design, the heartbeat table DDLs are ignored by the Extract and you should create the heartbeat tables manually at the target.

Installing DDL Replication

To install DDL replication, you run the installation script that is provided with Oracle GoldenGate as the replication user. This user must have `Create`, `Insert`,`Select`, `Delete`, `Drop`, and `Truncate` database privileges. Additionally, this user must have write permission to copy the Oracle GoldenGate plugin in the MySQL plugin directory. For example, the MySQL plugin are typically in `/usr/lib64/mysql/plugin/`.

The installation script options are `install`, `uninstall`, `start`, `stop`, and `restart`.

The command to install DDL replication uses the install option, user id, password, and port number respectively:

```
bash-3.2$ ./ddl_install.sh install-option user-id password port-number
```

For example:

```
bash-3.2$ ./ddl_install.sh install root welcome 3306
```

The DDL replication installation script completes the following tasks:

Using the Metadata Server

You can use the following options with the metadata server:

- You must have the Oracle GoldenGate `metadata_server` running to capture the DDL metadata.
- Run the install script with `start` option to start the metadata server.

- Run the install script with `stop` option to stop the metadata server.
- Run the install script with `restart` option to stop the running metadata server and start again.
- Oracle GoldenGate DDL replication uses two plugins as a shared library, `ddl_rewriter` and `ddl_metadata`, both of which must be installed on your MySQL server before Oracle GoldenGate replication starts.
- The `oggddl.history` metadata history table is used to store and retrieve the DDL metadata history.

There is a single history table and metadata server for each MySQL server. If you want to issue and capture DDLs from multiple instances of an Extract process on the same database server at the same time, there is a possibility of conflict between accessing and populating the metadata history table. Oracle recommends that you do not run and capture DDLs using multiple Extract instances on the same MySQL server.

Troubleshooting Plug-in Based DDL Replication

Plug-in based DDL replication relies on a metadata history table and the metadata plugin and server. To troubleshoot when DDL replication is enabled, the history table contents and the metadata plugin server logs are required.
You can use the `mysqldump` command to generate the history table dump using one of the following examples:

```
mysqldump [options] database [tables]
mysqldump [options] --databases [options] DB1 [DB2 DB3...]
mysqldump [options] --all-databases [options]
```

For example, `bash-3.2$ mysqldump -uroot -pwelcome oggddl history > outfile`

The metadata plugins and server logs are located in the MySQL and Oracle GoldenGate installation directories respectively.

If you find an error in the log files, you need to ensure that the metadata server is running.

Upgrading from Plugin-based DDL Replication to Transaction Log-based DDL Replication

If you are using the plug-in based solution on MySQL 5.7 and plan to upgrade to MySQL 8.0, which uses transaction log based DDL replication, you need to:

1. Uninstall the plug-in components as mentioned in Uninstalling Plug-In Based DDL Replication
2. Upgrade your database.
3. Re-enable DDL replication support based on the steps provided in MySQL: Prerequisites for Transaction Log Based DDL Configuration and check the prerequisites and configuration considerations.

Uninstalling Plug-In Based DDL Replication

If you no longer want to capture the DDL events, then you can use the same install script and select the `uninstall` option to disable the DDL setup. Also, any Extract with DDL parameters should be removed or disabled. If you want to capture the DDL again, you can run the install script again. You should take care when multiple instances of the capture process is running on the same instance of your MySQL server. The DDL setup should *not* be disturbed or

uninstalled when multiple capture processes are running and when at most one capture is designed to capture the DDL statement.

Use the installation script with the `uninstall` option to uninstall DDL Replication. For example:

```
bash-3.2$ ./ddl_install.sh uninstall root welcome 3306
```

The script performs the following tasks:

1. Uninstalls the `ddl_rewriter` and `ddl_metadata` plugins.
2. Deletes the `oggddl.history` table if exists.
3. Removes the plugins from MySQL plugin directory.
4. Stops the `metadata_server` if it is running.

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.*`	Include tables under the `ggvam` database.
`DDL INCLUDE OBJNAME atssrc.ggvam EXCLUDE OBJNAME atssrc.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'`	Include DDL that contains this string.
`DDL EXCLUDE INSTR 'WHY'`	Excludes DDL that contains this string.
`DDL INCLUDE MAPPED`	MySQL DDL uses this option and should be used as the default for Oracle GoldenGate MySQL DDL replication. `DDL INCLUDE ALL` and `DDL` are not supported.
`DDL EXCLUDE ALL`	Default option.

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:

  ```
  ddlops 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 `ddlops` specification).
- The semicolon is required to terminate the parameter entry.

  For example:

  ```
  ddl optype (create, drop), objname (eric.*);
  ddl exclude objname (eric.tab*);
  exttrail a;
  exttrail b;
  ddl optype (create), objname (joe.*), string ("abc", "xyz") local;
  ddl optype (alter), objtype (index);
  ```

  In this preceding example, the `exttrail a` gets creates and drops for all objects that belong to `eric`, except for objects that start with `tab`, `exttrail a` also gets all alter index statements, unless the index name begins with `tab` (the rule is global even though it's included in `exttrail b`). `exttrail b` gets the same objects as `a`, and it also gets all creates for objects that belong to `joe` when the string `abc`or `xyz` is present in the DDL text. The `ddlops.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.

Oracle: DDL Replication

Learn about DDL replication in Oracle.

Extract supports the DDL capture method for Oracle 11.2.0.4 or later. An Extract can capture DDL operations from a source Oracle database natively through the Oracle logmining server.

Prerequisites for Configuring DDL

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.

Oracle databases that have the COMPATIBLE parameter set to 11.2.0.4 or higher support DDL capture through the database logmining server. This method is known as native DDL capture. Native DDL capture is the only supported method for capturing DDL from a multitenant container database.

For downstream mining, the source database must also have database COMPATIBLE set to 11.2.0.4 or higher to support DDL capture through the database logmining server.

Overview of DDL Synchronization

Oracle GoldenGate supports the synchronization of DDL operations from one database to another.

DDL synchronization can be active when:

- business applications are actively accessing and updating the source and target objects.
- Oracle GoldenGate transactional data synchronization is active.

The components that support the replication of DDL and the replication of transactional data changes (DML) are independent of each other. Therefore, you can synchronize:

- just DDL changes
- just DML changes
- both DDL and DML

For a list of supported objects and operations for DDL support for Oracle, see Supported 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. Oracle GoldenGate 21c and higher releases support DDL statement length greater than 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.

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. For special considerations in an Oracle active-active configuration, see Propagating DDL in Active-Active (Bidirectional) Configurations.

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.

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.

Configuration Guidelines for DDL Support

Here are the guidelines for configuring Oracle GoldenGate processes to support DDL replication.

Topics:

Database Privileges

See Grant User Privileges for Oracle Database 21c and Lower.

Parallel Processing

If using parallel Extract and/or Replicat processes, keep related DDL and DML together in the same process stream to ensure data integrity. Configure the processes so that:

- all DDL and DML for any given object are processed by the same Extract group and by the same Replicat group.

- all objects that are relational to one another are processed by the same process group.

For example, if `repe` processes DML for `EMPLOYEES`, then it should also process the DDL for `EMPLOYEES`. If `APPRAISAL` has a foreign key to `EMPLOYEES`, then its DML and DDL operations also should be processed by `repe`.

If an Extract group writes to multiple trails that are read by different Replicat groups, Extract sends all of the DDL to all of the trails. Use each Replicat group to filter the DDL by using the filter options of the `DDL` parameter in the Replicat parameter file.

Object Names

Oracle GoldenGate preserves the database-defined object name, case, and character set. This support preserves single-byte and multibyte names, symbols, and accent characters at all levels of the database hierarchy.

Object names must be fully qualified with their two-part or three-part names when supplied as input to any parameters that support DDL synchronization. You can use the question mark (**?**) and asterisk (*****) wildcards to specify object names in configuration parameters that support DDL synchronization, but the wildcard specification also must be fully qualified as a two-part or three-part name. To process wildcards correctly, the `WILDCARDRESOLVE` parameter is set to `DYNAMIC` by default. If `WILDCARDRESOLVE` is set to anything else, the Oracle GoldenGate process that is processing DDL operations will abend and write the error to the process report.

Data Definitions

Because DDL support requires a like-to-like configuration, the `ASSUMETARGETDEFS` parameter must be used in the Replicat parameter file. Replicat will abend if objects are configured for DDL support and the `SOURCEDEFS` parameter is being used.

For more information about `ASSUMETARGETDEFS`, see *Parameters and Functions Reference for Oracle GoldenGate*.

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 fin2 and move new or renamed tables into that schema, the parameter statements could look as follows, depending on whether you want the fin2 objects mapped to the same, or different, schema on the target:

Extract:

```
TABLE fin2.*;
```

Replicat:

```
MAP fin2.*, 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.

Topics:

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 fin.expen;
TABLE hr.tab*;
```

Replicat (target)

```
MAP fin.expen, TARGET fin2.expen2;
MAP hr.tab*, TARGET hrBackup.bak_*;
```

Also assume a source DDL statement of:

```
ALTER TABLE fin.expen 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 fin2.expen2 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.

Topics:

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 logmining 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`

Topics:

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.

Controlling the Propagation of DDL to Support Different Topologies

To support bidirectional and cascading replication configurations, it is important for Extract to be able to identify the DDL that is performed by Oracle GoldenGate and by other applications, such as the local business applications.

Depending on the configuration that you want to deploy, it might be appropriate to capture one or both of these sources of DDL on the local system.

> **Note:**
>
> Oracle GoldenGate DDL consists of ALTER TABLE statements performed by Extract to create log groups and the DDL that is performed by Replicat to replicate source DDL changes.

The following options of the DDLOPTIONS parameter control whether DDL on the local system is captured by Extract and then sent to a remote system, assuming Oracle GoldenGate DDL support is configured and enabled:

- The GETREPLICATES and IGNOREREPLICATES options control whether Extract captures or ignores the DDL that is generated by Oracle GoldenGate. The default is IGNOREREPLICATES, which does not propagate the DDL that is generated by Oracle GoldenGate. To identify the DDL operations that are performed by Oracle GoldenGate, the following comment is part of each Extract and Replicat DDL statement:

  ```
  /* GOLDENGATE_DDL_REPLICATION */
  ```

- The GETAPPLOPS and IGNOREAPPLOPS options control whether Extract captures or ignores the DDL that is generated by applications other than Oracle GoldenGate. The default is GETAPPLOPS, which propagates the DDL from local applications (other than Oracle GoldenGate).

The result of these default settings is that Extract ignores its own DDL and the DDL that is applied to the local database by a local Replicat, so that the DDL is not sent back to its source,

and Extract captures all other DDL that is configured for replication. The following is the default `DDLOPTIONS` configuration.

```
DDLOPTIONS GETAPPLOPS, IGNOREREPLICATES
```

This behavior can be modified. See the following topics:

Propagating DDL in Active-Active (Bidirectional) Configurations

Oracle GoldenGate supports active-active DDL replication between two systems. For an active-active bidirectional replication, the following must be configured in the Oracle GoldenGate processes:

1. DDL that is performed by a business application on one system must be replicated to the other system to maintain synchronization. To satisfy this requirement, include the `GETAPPLOPS` option in the `DDLOPTIONS` statement in the Extract parameter files on both systems.
2. DDL that is applied by Replicat on one system must be captured by the local Extract and sent back to the other system. To satisfy this requirement, use the `GETREPLICATES` option in the `DDLOPTIONS` statement in the Extract parameter files on both systems.

   > **Note:**
   >
   > An internal Oracle GoldenGate token will cause the actual Replicat DDL statement itself to be ignored to prevent loopback. The purpose of propagating Replicat DDL back to the original system is so that the Replicat on that system can update its object metadata cache, in preparation to receive incoming DML, which will have the new metadata.

3. Each Replicat must be configured to update its object metadata cache whenever the remote Extract sends over a captured Replicat DDL statement. To satisfy this requirement, use the `UPDATEMETADATA` option in the `DDLOPTIONS` statement in the Replicat parameter files on both systems.

The resultant `DDLOPTIONS` statements should look as follows:

Extract (primary and secondary)

```
DDLOPTIONS GETREPLICATES, GETAPPLOPS
```

Replicat (primary and secondary)

```
DDLOPTIONS UPDATEMETADATA
```

> **WARNING:**
>
> Before you allow new DDL or DML to be issued for the same object(s) as the original DDL, allow time for the original DDL to be replicated to the remote system and then captured again by the Extract on that system. This will ensure that the operations arrive in correct order to the Replicat on the original system, to prevent DML errors caused by metadata inconsistencies. See the following diagram for more information.

For more information, see *Parameters and Functions Reference for Oracle GoldenGate*.

Prerequisites for Configuring DDL

In a cascading configuration, use the following setting for `DDLOPTIONS` in the Extract parameter file on each intermediary system. This configuration forces Extract to capture the DDL from Replicat on an intermediary system and cascade it to the next system downstream.

```
DDLOPTIONS GETREPLICATES, IGNOREAPPLOPS
```

For more information about `DDLOPTIONS`, see `DDLOPTIONS`.

Add Supplemental Log Groups Automatically

Use the `DDLOPTIONS` parameter with the `ADDTRANDATA` option for performing tasks described in this topic.

You can perform the following tasks using the `DDLOPTIONS`:

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

To use `DDLOPTIONS ADDSCHEMATRANDATA`, the `ADD SCHEMATRANDATA` command must be issued on the Admin Client to enable schema-level supplemental logging.

By default, the `ALTER TABLE` that adds the supplemental logging is not replicated to the target unless the `GETREPLICATES` parameter is in use.

`DDLOPTIONS ADDTRANDATA` is not supported for multitenant container databases, see Configure 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 10*g* or 11*g* 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
```

Topics:

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:

```
2011-01-20 15:11:45  GGS INFO     2104  DDL found, operation [drop table
myTableTemp ], Source SCN [1186713.0].
 2011-01-20 15:11:45  GGS INFO     2100  DDL is of mapped scope, after
mapping new operation [drop table "QATEST2"."MYTABLETEMP" ].
 2011-01-20 15:11:45  GGS INFO     2100  DDL operation included [include
objname myTable*], optype [DROP], objtype [TABLE], objname
[QATEST2.MYTABLETEMP].
 2011-01-20 15:11:45  GGS INFO     2100  Executing DDL operation.
 2011-01-20 15:11:48  GGS INFO     2105  DDL error ignored for next retry:
error code [942], filter [include objname myTableTemp], error text
[ORA-00942: table or view does not exist], retry [1].
 2011-01-20 15:11:48  GGS INFO     2100  Executing DDL operation , trying
again due to RETRYOP parameter.
 2011-01-20 15:11:51  GGS INFO     2105  DDL error ignored for next retry:
error code [942], filter [include objname myTableTemp], error text
[ORA-00942: table or view does not exist], retry [2].
 2011-01-20 15:11:51  GGS INFO     2100  Executing DDL operation, trying
again due to RETRYOP parameter.
 2011-01-20 15:11:54  GGS INFO     2105  DDL error ignored for next retry:
error code [942], filter [include objname myTableTemp], error text
[ORA-00942: table or view does not exist], retry [3].
 2011-01-20 15:11:54  GGS INFO     2100  Executing DDL operation, trying
again due to RETRYOP parameter.
 2011-01-20 15:11:54  GGS INFO     2105  DDL error ignored: error code [942],
filter [include objname myTableTemp], error text [ORA-00942: table or view
does not exist].
```

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, 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
myTable (
    myId number (10) not null,
    myNumber number,
    myString varchar2(100),
    myDate date,
    primary key (myId)
```

```
) ], 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 myTable*], optype [CREATE], objtype [TABLE], objname
[QATEST1.MYTABLE].

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 TRAN DATA for
table with the key, table [QATEST1.MYTABLE], operation [ALTER TABLE
"QATEST1"."MYTABLE" ADD SUPPLEMENTAL LOG GROUP "GGS_MYTABLE_53475" (MYID)
ALWAYS  /* GOLDENGATE_DDL_REPLICATION */ ].

2011-01-20 15:11:43  GGS INFO     2100  DDL found, operation [create table
myTableTemp (
    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 myTableTemp OPTYPE CREATE], optype [CREATE], objtype [TABLE], objname
[QATEST1.MYTABLETEMP].
```

Statistics in the Process Reports

You can send current statistics for DDL processing to the Extract and Replicat reports by using the `SEND` command in Admin Client.

```
SEND {EXTRACT | REPLICAT} group REPORT
```

The statistics show totals for:

- All DDL operations
- Operations that are `MAPPED` in scope
- Operations that are `UNMAPPED` in scope
- Operations that are `OTHER` in scope
- Operations that were excluded (number of operations minus included ones)
- Errors (Replicat only)
- Retried errors (Replicat only)
- Discarded errors (Replicat only)
- Ignored operations (Replicat only)

Tracing DDL Processing

If you open a support case with Oracle GoldenGate Technical Support, you might be asked to turn on tracing. `TRACE` and `TRACE2` control DDL tracing.

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 does not actually 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.

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:

```
0000000000000000001:f77024f9-f4e3-11eb-a052-0021f6e03f10:0000000000000010654
```

In this sequence, the Oracle GoldenGate sequence number is `0000000000000000001` 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 type `positiob` by GTID set 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": "ggeast"
    },
    "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
0000000000000000000: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 and higher and requires Oracle GoldenGate version 21.7 or higher.
- Only Group Replication configured in Single-Primary Mode is supported for Extract.
- The MySQL database setting `gtid_mode` must be enabled.

Topic:

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 Extract with MySQL 21.7 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 RMTTRAILwith 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.

  If you need to use remote trails, then you can use data Pump to send the trail in Classic Architecture. In Microservices Architecture, use the `DISTPATH` to transport the trail. See Manage Distribution Paths.

SSL Configuration on Group Replication Cluster

Learn about SSL configuration on Group Replication Cluster.

Topics:

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 11-1 Router and Database Node Configuration

Router	**-**
Hostname	`mysqlrouter.company.com`
Config Filename	`mysqlrouter.conf`
Port	`6446`
Common Name	`mysqlrouter.company.com`
Certificate Name	`server-cert.pem`
Key file name	`server-key.pem`
Database Node 1	-
Hostname	`dbnode1.company.com`
Config Filename	`my.cnf`
Port	`3308`
Common Name	`dbnode1`
Certificate Name	`server-cert.pem`
Key file name	`server-key.pem`
Node Rank	Primary
Database Node2	-
Hostname	`dbnode2.company.com`
Config Filename	`my.cnf`
Port	`3308`
Common Name	`dbnode2`
Certificate Name	`server-cert.pem`

Table 11-1 (Cont.) Router and Database Node Configuration

Router	**-**
Key file name	`server-key.pem`
Node Rank	Secondary
Database Node3	-
Hostname	`dbnode3.company.com`
Config Filename	`my.cnf`
Port	`3308`
Common Name	`dbnode3`
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 11-1 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 11-1 Running the `GG_PROCEDURE_REPLICATION_ON` Function

```
SET SERVEROUTPUT ON
DECLARE
  on_or_off   NUMBER;
BEGIN
  on_or_off := DBMS_GOLDENGATE_ADM.GG_PROCEDURE_REPLICATION_ON;
  IF on_or_off=1 THEN
    DBMS_OUTPUT.PUT_LINE('Oracle GoldenGate procedural replication is ON.');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Oracle GoldenGate procedural replication is OFF.');
  END IF;
END;
/
```

Enabling and Disabling Supplemental Logging

Oracle GoldenGate provides commands to allow you to enable or disable procedural supplemental logging.

To enable supplemental logging:

1. Connect to the source database as the Oracle GoldenGate administrator with `DBLOGIN`.

   ```
   CONNECT https://localhost:9000 DEPLOYMENT demo AS ggadmin PASSWORD adminpw


   DBLOGIN USERIDALIAS ggeast DOMAIN OracleGoldenGate
   ```

2. Add supplemental logging for procedural replication.

   ```
   ADD PROCEDURETRANDATA
   ```

   The output shows:

   ```
   INFO OGG-13005 PROCEDURETRANDATA supplemental logging has been enabled.
   ```

Supplemental logging is enabled for procedure replication.

To disable supplemental logging:

1. Connect to the source database as the Oracle GoldenGate administrator with dblogin.

```
CONNECT https://localhost:9000 DEPLOYMENT demo AS ggadmin PASSWORD adminpw


DBLOGIN USERIDALIAS ggeast DOMAIN OracleGoldenGate
```

2. Remove supplemental logging for procedure replication.

```
DELETE PROCEDURETRANDATA
```

Supplemental logging is disabled for procedure replication.

To view information about supplemental logging:

1. Connect to the source database as the Oracle GoldenGate administrator with dblogin.

```
CONNECT https://localhost:9000 DEPLOYMENT demo AS ggadmin PASSWORD adminpw


DBLOGIN USERIDALIAS ggeast DOMAIN OracleGoldenGate
```

2. Display supplemental logging information for procedure replication.

```
INFO PROCEDURETRANDATA
```

Supplemental logging information for procedure replication is displayed.

Filtering Features for Procedural Replication

You can specify which procedures and packages you want to include or exclude for procedure replication.

You group supported packages and procedures using feature groups. You use the procedure parameter with the `INCLUDE` or `EXCLUDE` keyword to filter features for procedure replication.

In the procedure parameter, `INCLUDE` or `EXCLUDE` specify the beginning of a filtering clause. They specify the procedures to replicate (`INCLUDE`) or filter out (`EXCLUDE`). The filtering clause must consist of the `INCLUDE ALL_SUPPORTED` or `EXCLUDE ALL_SUPPORTED` keyword followed by any valid combination of the other filtering options of the procedure parameter. The `EXCLUDE` filter takes precedence over any `INCLUDE` filters that contain the same criteria.

> **Note:**
>
> When replicating Oracle Streams Advanced Queuing (AQ) procedures, you must use the `RULE` option in your parameter file as follows:
>
> ```
> PROCEDURE INCLUDE FEATURE ALL_SUPPORTED
> ```
>
> or
>
> ```
> PROCEDURE INCLUDE FEATURE AQ, RULE
> ```
>
> Do *not* use `PROCEDURE INCLUDE FEATURE AQ` without the `RULE` option.

Including all system supplied packages at Extract:

1. Connect to Extract in the source database.

   ```
   EXTRACT edba
   USERIDALIAS admin_dbA DOMAIN ORADEV
   ```

2. Create a new trail file.

   ```
   EXTTRAIL ea
   ```

3. Enable procedure replication, if not already done.

   ```
   TRANLOGOPTIONS INTEGRATEDPARAMS (ENABLE_PROCEDURAL_REPLICATION Y)
   ```

4. Include filter for procedure replication.

   ```
   PROCEDURE INCLUDE FEATURE ALL_SUPPORTED
   ```

You have successfully included all system supplied packages for procedure replication.

Excluding specific packages at Replicat:

1. Connect to Replicat in the target database.

   ```
   REPLICAT rdba
   USERIDALIAS admin_dbBDOMAIN ORADEV
   ```

2. Include filter for procedure replication.

   ```
   PROCEDURE EXCLUDE FEATURE RLS
   ```

You have successfully excluded specific packages for procedure replication.

Handling Procedural Replication Errors

Procedural replication uses `REPERROR` parameter to configure the behavior of Replicat when an procedural error occurs.

By default, Replicat will abend when a procedural replication occurs so using the following steps sets up error handling:

1. Connect to Replicat in the target database.

   ```
   REPLICAT rdba
   USERIDALIAS admin_dbBDOMAIN ORADEV
   ```

2. Include filter for procedure replication.

```
PROCEDURE EXCLUDE FEATURE RLS
```

3. Specify error handling parameter, see REPERROR in *Parameters and Functions Reference for Oracle GoldenGate* for other options.

```
REPERROR (PROCEDURE, DISCARD)
```

You have successfully handled errors for procedural replication.

Listing the Procedures Supported for Oracle GoldenGate Procedural Replication

The `DBA_GG_SUPPORTED_PROCEDURES` view displays information about the supported packages for Oracle GoldenGate procedural replication.

When a procedure is supported and Oracle GoldenGate procedural replication is on, calls to the procedure are replicated, unless the procedure is excluded specifically.

1. Connect to the database as `sys` (`sqlplus`, `sqlcl`, `sqldeveloper`) not as an Oracle GoldenGate administrator.
2. Query the `DBA_GG_SUPPORTED_PROCEDURES` view.

Example 11-2 Displaying Information About the Packages Supported for Oracle GoldenGate Procedural Replication

This query displays the following information about the packages:

- The owner of each package
- The name of each package
- The name of each procedure
- The minimum database release from which the procedure is supported
- Whether there is an exclusion rule that prevents the procedure from being replicated for some database objects

```
COLUMN OWNER FORMAT A10
COLUMN PACKAGE_NAME FORMAT A15
COLUMN PROCEDURE_NAME FORMAT A15
COLUMN MIN_DB_VERSION FORMAT A14
COLUMN EXCLUSION_RULE_EXISTS FORMAT A14

SELECT OWNER,
       PACKAGE_NAME,
       PROCEDURE_NAME,
       MIN_DB_VERSION,
       EXCLUSION_RULE_EXISTS
  FROM DBA_GG_SUPPORTED_PROCEDURES;
```

Your output looks similar to the following:

```
OWNER      PACKAGE_NAME    PROCEDURE_NAME  MIN_DB_VERSION EXCLUSION_RULE
---------- --------------- --------------- -------------- --------------
XDB        DBMS_XDB_CONFIG ADDTRUSTMAPPING 12.2           NO
CTXSYS     CTX_DDL         ALTER_INDEX     12.2           NO
SYS        DBMS_FGA        DROP_POLICY     12.2           NO
SYS        XS_ACL          DELETE_ACL      12.2           NO
.
```

.
.

Monitoring Oracle GoldenGate Procedural Replication

A set of data dictionary views enable you to monitor Oracle GoldenGate procedural replication.

You can use the following views to monitor Oracle GoldenGate procedural replication:

View	Description
DBA_GG_SUPPORTED_PACKAGES	Provides details about supported packages for Oracle GoldenGate procedural replication. When a package is supported and Oracle GoldenGate procedural replication is on, calls to subprograms in the package are replicated.
DBA_GG_SUPPORTED_PROCEDURES	Provides details about the procedures that are supported for Oracle GoldenGate procedural replication.
DBA_GG_PROC_OBJECT_EXCLUSION	Provides details about all database objects that are on the exclusion list for Oracle GoldenGate procedural replication. A database object is added to the exclusion list using the INSERT_PROCREP_EXCLUSION_OBJ procedure in the DBMS_GOLDENGATE_ADM package. When a database object is on the exclusion list, execution of a subprogram n 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.

Configure Managed 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 Profile from the Administration Service navigation pane.
2. In the Managed Process Settings tab, you can click + sign to start creating a profile. There's also a default profile preset on this page.
3. Enter the details for the profile options including the Profile Name, Description, Auto Start and Auto Restart options. See the following table for Auto Start and Auto Restart options.

Option	Description
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 failes
Disable Task After Retries Exhausted	If true then the task is disabled after exhausting all attempts to restart the process.

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 procedure_name()'`	Execute a stored procedure
`SQLEXEC 'sql_query'`	Execute a query
`SQLEXEC 'database_command'`	Execute a database command

Argument	Description
`'call procedure_name ()'`	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.
sp_name	Specifies the name of the stored procedure to execute.
`ID logical_name`	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.

Argument	Description
PARAMS *param_spec* | 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 *logical_name*	Defines a logical name for the query. A logical name is required in order to extract values from the query results. ID *logical_name* references the column values returned by the query.
QUERY ' *sql_query* '	Specifies the SQL query syntax to execute against the database. 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"'`
PARAMS *param_spec* | 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 parametes. See SQLEXEC in the *Parameters and Functions Reference for Oracle GoldenGate*.

Example 11-3 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 11-4 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 11-2 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.

Topics:

Guidelines for Using Self-describing Trails

Self-describing trail files are the default if the trail file format is 12.2 or higher, if you are not using SOURCEDEFS OVERRIDE or TARGETDEFS OVERRIDE. Oracle recommends that you use self-

describing trail files. You should only use SOURCEDEFS OVERRIDE and TARGETDEFS OVERRIDE for backward compatibility requirements.

The following are the guidelines for using self-describing trails:

- 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.
- If the source Oracle GoldenGate release is 12.1 or earlier, then you need to use SOURCEDEFS OVERRIDE or TARGETDEFS OVERRIDE. See SOURCEDEFS OVERRIDE and TARGETDEFS OVERRIDE in the *Parameters and Functions Reference for Oracle GoldenGate*.

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.

Topics:

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 Eequence

- 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:

Topics:

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.

Topics:

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 11-3 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 ("ColA" = ColA)
```

See Specifying Object Names in Oracle GoldenGate Input for more information about specifying names to Oracle GoldenGate.

See Globalization Considerations when Mapping Data for globalization considerations when mapping source and target columns in databases that have different character sets and locales.

Avoid using `COLMAP` to map a value to a key column (which causes the operation to become a primary key update), The `WHERE` clause that Oracle GoldenGate uses to locate the target row will not use the correct before image of the key column. Instead, it will use the after image. This will cause errors if you are using any functions based on that key column, such as a `SQLEXEC` statement.

Column Mapping Limitations

Here are the column mapping limitations:

- LOB columns cannot be used in FILTER, WHERE 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
NAMES *target_column* = *source_column*	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`

PREFIX *prefix* | SUFFIX *suffix*	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"`

RESET	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

The following explains how Oracle GoldenGate maps data types.

Topics:

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.FFFFFF`.

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.

Topics:

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.TARGTAB,
                  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 11-4 Using Multiple FILTER Statements

Parameter file	Description
`REPERROR (9999, EXCEPTION)`	Raises an exception for the specified error.
`MAP OWNER.SRCTAB,` `TARGET OWNER.TARGTAB,`	Starts the MAP statement.

`SQLEXEC (ID CHECK, ON UPDATE,`
`QUERY ' SELECT COUNT FROM TARGTAB '`
`'WHERE PKCOL = :P1 ',`
`PARAMS (P1 = PKCOL)),`
	Performs a query to retrieve the present value of the COUNT column whenever an update is encountered. There is a BEFOREFILTER option also that allows the query or stored procedure to be executed prior to processing the FILTER clause. This allows values from the SQLEXEC portion to be used inside the FILTER at runtime.

`FILTER (BALANCE > 15000),`	Uses a FILTER clause to select rows where the balance is greater than 15000.
`FILTER (ON UPDATE, @BEFORE (COUNT) =` `CHECK.COUNT)`	Uses another FILTER clause to ensure that the value of the source COUNT column before an update matches the value in the target column before applying the target update.
`;`	The semicolon concludes the MAP statement.

Table 11-4 (Cont.) Using Multiple FILTER Statements

Parameter file	Description

`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 11-5 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 11-6 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 11-7 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 11-8 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 11-4, 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 11-5 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 11-5 Permissible WHERE Operators

Element	Examples
Column names	`PRODUCT_AMT`
Numeric values	`-123, 5500.123`
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 11-6 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 11-6 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	`GETUPDATEBEFORES` `TABLE ACCOUNT.CUSTOMER;`

Replicat	`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);`

> **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 11-7 Conversion Function Syntax

Syntax element	Description
@*function*	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.
argument	A function argument.

Table 11-8 Function Arguments

Argument element	Example
A numeric constant	`123`
A string literal enclosed within single quote marks	`'ABCD'`

Table 11-8 (Cont.) Function Arguments

Argument element	Example
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 11-9 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 11-9 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
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

Topics:

Defining Tokens

To use tokens, you define the token name and associate it with data. The data can be any valid character data or values retrieved from Oracle GoldenGate column-conversion functions.

The token area in the record header permits up to 16,000 bytes of data. Token names, the length of the data, and the data itself must fit into that space.

To define a token, use the `TOKENS` option of the `TABLE` parameter in the Extract parameter file.

Syntax

```
TABLE table_spec, TOKENS (token_name = token_data [, ...]);
```

Where:

- *`table_spec`* is the name of the source table. A container or catalog name, if applicable, and an owner name must precede the table name.
- *`token_name`* is a name of your choice for the token. It can be any number of alphanumeric characters and is not case-sensitive.
- *`token_data`* is a character string of up to 2000 bytes. The data can be either a string that is enclosed within single quotes or the result of an Oracle GoldenGate column-conversion function. The character set of token data is not converted. The token must be in the character set of the source database for Extract and in the character set of the target database for Replicat. In the trail file, user tokens are stored in UTF-8.

```
TABLE ora.oratest, TOKENS (
TK-OSUSER = @GETENV ('GGENVIRONMENT' , 'OSUSERNAME'),
TK-GROUP = @GETENV ('GGENVIRONMENT' , 'GROUPNAME')
TK-HOST =  @GETENV('GGENVIRONMENT' , 'HOSTNAME'));
```

As shown in this example, the Oracle GoldenGate `@GETENV` function is an effective way to populate token data. This function provides several options for capturing environment information that can be mapped to tokens and then used on the target system for column mapping.

Using Token Data in Target Tables

To map token data to a target table, use the `@TOKEN` column-conversion function in the source expression of a `COLMAP` clause in a Replicat `MAP` statement. The `@TOKEN` function provides the name of the token to map. The `COLMAP` syntax with `@TOKEN` is:

Syntax

```
COLMAP (target_column = @TOKEN ('token_name'))
```

The following `MAP` statement maps target columns `host`, `gg_group`, and so forth to tokens `tk-host`, `tk-group`, and so forth. Note that the arguments must be enclosed within single quotes.

User tokens	Values
tk-host	:sysA
tk-group	:extora
tk-osuser	:jad
tk-domain	:admin

User tokens	Values
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 11-10 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.

Topics:

Prerequisites for Bidirectional Replication

Topics:

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 FILTERTABLE` parameter for the CDC Extract. The table used as the filtering table will be the Oracle GoldenGate checkpoint table that you must create for the Replicat.

> **Note:**
>
> Only Classic and Coordinated Replicats support bi-directional and multi-directional replication, Parallel Replicat does not support this.

To create a Filter Table and enable Supplemental Logging:

1. On each source database, ensure that a checkpoint table for use by Replicats has been created. For example:

   ```
   ADD CHECKPOINTTABLE ggadmin.oggcheck
   ```

2. Enable supplemental logging for the the checkpoint table. For example:

   ```
   ADD TRANDATA ggadmin.ggcheckpoint ALLCOLS
   ```

3. Ensure that the Replicat is created with the checkpoint table information.

   ```
   ADD REPLICAT reptgt1, EXTTRAIL ./dirdat/e2, CHECKPOINTTABLE
   ggadmin.ggcheckpoint
   ```

4. Configure each Extract with the `IGNOREREPLICATES` (on by default) and `FILTERTABLE` parameters, using the Replicat's checkpoint table for the filtering table.

   ```
   TRANLOGOPTIONS IGNOREREPLICATES TRANLOGOPTIONS FILTERTABLE
   ggadmin.ggcheckpoint
   ```

   > **Note:**
   >
   > Oracle GoldenGate for PostgreSQL supports only one `FILTERTABLE` 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.

Topics:

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 FILTERTABLE 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. `FILTERTABLE` 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 FILTERTABLE 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. `FILTERTABLE` identifies the name of the checkpoint table, so that Extract ignores transactions that contain any operations on it.

Oracle

There are multiple ways to identify Replicat transaction in an Oracle environment. When Replicat is in classic or integrated mode, you use the following parameters:

- Replicats set a tag of 00 by default. Use `DBOPTIONS` with the `SETTAG` option in the Replicat parameter file to change the tag that Replicat sets. Replicat tags the transactions being applied with the specified value, which identifies those transactions in the redo stream. Valid values are a single TAG value consisting of hexadecimal digits.

- Use the TRANLOGOPTIONS parameter with the EXCLUDETAG option in the Extract parameter file. The logmining server associated with that Extract excludes redo that is tagged with the SETTAG value.

  The following shows how SETTAG can be set in the Replicat parameter file:

  ```
  DBOPTIONS SETTAG 0935
  ```

  The following shows how EXCLUDETAG can be set in the Extract parameter file:

  ```
  TRANLOGOPTIONS EXCLUDETAG 0935
  ```

  If you are excluding multiple tags, each must have a separate TRANLOGOPTIONS EXCLUDETAG statement specified.

You can also use the transaction name or USERID of the Replicat user to identify Replicat transactions. You can choose which of these to ignore when you configure Extract.

Preventing the Capture of Replicat Operations

Depending on which database you are using, you may or may not need to provide explicit instructions to prevent the capture of Replicat operations.

Oracle: Preventing the Capture of Replicat Transactions

To prevent the capture of SQL that is applied by Replicat to an Oracle database, use the TRANLOGOPTIONS parameter with the EXCLUDETAG *tag* option. This parameter directs the Extract process to ignore transactions that are tagged with the specified redo tag.

See Identifying Replicat Transactions to set the tag value. This is the recommended approach for Oracle.

Non-Oracle Database: Preventing Capture of Replicat Transactions

To prevent the capture of SQL that is applied by Replicat to other database types, use the following parameters:

- GETAPPLOPS | IGNOREAPPLOPS: Controls whether or not data operations (DML) produced by business applications *except Replicat* are included in the content that Extract writes to a specific trail or file.
- GETREPLICATES | IGNOREREPLICATES: Controls whether or not DML operations produced by *Replicat* are included in the content that Extract writes to a specific trail or file.

Manage Conflicts

Uniform conflict-resolution procedures must be in place on all systems in an active-active configuration. Conflicts should be identified immediately and handled with as much automation as possible; however, different business applications will present their own unique set of requirements in this area.

Because Oracle GoldenGate is an asynchronous solution, conflicts can occur when modifications are made to identical sets of data on separate systems at (or almost at) the same time. Conflicts occur when the timing of simultaneous changes results in one of these out-of-sync conditions:

- A **uniqueness conflict** occurs when Replicat applies an insert or update operation that violates a uniqueness integrity constraint, such as a PRIMARY KEY or UNIQUE constraint. An example of this conflict type is when two transactions originate from two different databases, and each one inserts a row into a table with the same primary key value.

- An **update conflict** occurs when Replicat applies an update that conflicts with another update to the same row. Update conflicts happen when two transactions that originate from different databases update the same row at nearly the same time. Replicat detects an update conflict when there is a difference between the old values (the before values) that are stored in the trail record and the current values of the same row in the target database.
- A **delete conflict** occurs when two transactions originate at different databases, and one deletes a row while the other updates or deletes the same row. In this case, the row does not exist to be either updated or deleted. Replicat cannot find the row because the primary key does not exist.

For example, UserA on DatabaseA updates a row, and UserB on DatabaseB updates the same row. If UserB's transaction occurs before UserA's transaction is synchronized to DatabaseB, there will be a conflict on the replicated transaction.

A more complicated example involves three databases and illustrates a more complex ordering conflict. Assume three databases A, B, and C. Suppose a user inserts a row at database A, which is then replicated to database B. Another user then modifies the row at database B, and the row modification is replicated to database C. If the row modification from B arrives at database C before the row insert from database A, C will detect a conflict.

Where possible, try to minimize or eliminate any chance of conflict. Some ways to do so are:

- Configure the applications to restrict which columns can be modified in each database. For example, you could limit access based on geographical area, such as by allowing different sales regions to modify only the records of their own customers. As another example, you could allow a customer service application on one database to modify only the `NAME` and `ADDRESS` columns of a customer table, while allowing a financial application on another database to modify only the `BALANCE` column. In each of those cases, there cannot be a conflict caused by concurrent updates to the same record.
- Keep synchronization latency low. If UserA on DatabaseA and UserB on DatabaseB both update the same rows at about the same time, and UserA's transaction gets replicated to the target row before UserB's transaction is completed, conflict is avoided. See Managing Conflicts for suggestions on improving the performance of the Oracle GoldenGate processes.

To avoid conflicts, replication latency must be kept as low as possible. When conflicts are unavoidable, they must be identified immediately and resolved with as much automation as possible, either through the Oracle GoldenGate Conflict Detection and Resolution (CDR) feature, or through methods developed on your own. Custom methods can be integrated into Oracle GoldenGate processing through the `SQLEXEC` and user exit functionality. See Manual Conflict Detection and Resolution for more information about using Oracle GoldenGate to handle conflicts.

For Oracle database, the automatic Conflict Detection Resolution (CDR) feature exists. To know more, see Automatic Conflict Detection and Resolution.

MySQL: Bi-Directional Replication

In a bidirectional configuration, there are Extract and Replicat processes on both the source and target systems to support the replication of transactional changes on each system to the other system. To support this configuration, each Extract must be able to filter the transactions applied by the local Replicat, so that they are not recaptured and sent back to their source in a continuous loop. Additionally, `AUTO_INCREMENT` columns must be set so that there is no conflict between the values on each system.

1. Configure Oracle GoldenGate for high availability or active-active replication according to the instructions in the Propagating DDL in Active-Active (Bidirectional) Configurations.

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 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 `FILTERTABLE` 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 FILTERTABLE` option. Multiple filter tables allow the `TRANLOGOPTIONS FILTERTABLE` to be specified multiple times with different table names or wildcards.

     You can include single or multiple `TRANLOGOPTIONS FILTERTABLE` entries in the Extract parameter file. In the following example, multiple `TRANLOGOPTIONS FILTERTABLE`entries are included in the Extract parameter file with explicit object names and wildcards.

     ```
     TRANLOGOPTIONS FILTERTABLE ggs.chkpt2
     TRANLOGOPTIONS FILTERTABLE ggs.chkpt_RABC_*
     ```

3. 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 Propagating DDL in Active-Active (Bidirectional) Configurations.
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 `FILTERTABLE` 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 FILTERTABLE` option. Multiple filter tables allow the `TRANLOGOPTIONS FILTERTABLE` to be specified multiple times with different table names or wildcards.

     You can include single or multiple `TRANLOGOPTIONS FILTERTABLE` entries in the Extract parameter file. In the following example, multiple `TRANLOGOPTIONS FILTERTABLE`entries are included in the Extract parameter file with explicit object names and wildcards.

     ```
     TRANLOGOPTIONS FILTERTABLE ggs.chkpt2
     TRANLOGOPTIONS FILTERTABLE 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.

Topics:

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.

The procedures that follow assume that Oracle GoldenGate is configured properly to support active-active configuration. This means that it must be:

- Installed according to the instructions in this guide.
- Configured according to the instructions in the Oracle GoldenGate Windows and UNIX Administrator's Guide.

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

   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 11-11

```
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 11-12 Node 1

```
DECLARE
DBMS_DBFS_CONTENT.MOUNTSTORE('FS1', 'LOCAL');
DBMS_DBFS_CONTENT.MOUNTSTORE('FS2', 'REMOTE',
READ_ONLY => TRUE);
COMMIT;
END;
/
```

Example 11-13 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 11-14

```
select fs.store_name, tb.table_name, tb.ptable_name
from table(dbms_dbfs_sfs.listTables) tb,
table(dbms_dbfs_sfs.listfile systems) fs
where   fs.schema_name = tb.schema_name
and fs.table_name = tb.table_name
and fs.store_name in ('FS1', 'FS2')
;
```

Example 11-15 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 11-16 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 11-17 Node1

```
TABLE [container.]schema.SFS$_FST_100
TABLE [container.]schema.SFS$_FSTP_100;
```

Example 11-18 Node2

```
TABLE [container.]schema.SFS$_FST_119
TABLE [container.]schema.SFS$_FSTP_119;
```

Example 11-19 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 11-20 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.

Also see: Monitor.

Automatic Conflict Detection and Resolution

When Oracle GoldenGate replicates changes between Oracle databases, you can configure and manage Oracle GoldenGate automatic conflict detection and resolution in the Oracle databases.

> **Note:**
>
> The automatic conflict detection and resolution feature is specific to Oracle Database 12c Release 2 (12.2) and later, which is configured in an Oracle database. It also requires Oracle GoldenGate 12c (12.3.0.1) and later. 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

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.

Topics:

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. These are administration APIs, which are expected to be called by an Oracle GoldenGate administrator who is granted privileges through the `DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE`. This user needs to have privileges to modify the affected table. These APIs result in the version number being bumped for the object and also lock the object due to DDL execution.

The administrator user must be logged in to the appropriate PDB when calling these APIs. Three constants, which represent bit flags are now added:

- `EARLIEST_TIMESTAMP_RESOLUTION=0x0001` sets `TOMBSTONE KEY VERSIONING` automatically
- `DELETE_ALWAYS_WINS=0x0002` sets `TOMBSTONE KEY VERSIONING` automatically.
- `IGNORE_SITE_PRIORITY=0x0004`

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.

> **See Also:**
>
> - *Oracle Database Utilities* for information about supplemental logging

Requirements for Automatic Conflict Detection and Resolution

Supplemental logging is required to ensure that each row LCR has the information required to detect and resolve a conflict. Supplemental logging places additional information in the redo log for the columns of a table when a DML operation is performed on the table. When you configure a table for Oracle GoldenGate conflict detection and resolution, supplemental logging is configured automatically for all of the columns in the table. The additional information in the redo log is placed in an LCR when a table change is replicated.

Extract must be used for capturing. Integrated Replicat or parallel Replicat in integrated mode must be used on the apply side. `LOGALLSUPCOLS` should remain the default.

There is a hidden field `KEYVER$$` of type timestamp that is optionally added to the `DELETE TOMBSTONE` table. This field is required for `EARLIEST TIMESTAMP`, `DELETE ALWAYS WINS`, and `SITE PRIORITY` resolution and it also exists in the base table. The existence of the field in the base table needs to be provided in the trail file metadata as a flag or token.

Primary Key updates is also supported in the `DELETE TOMBSTONE` table. An entry is inserted into the `DELETE TOMBSTONE` table for the row of the original key value (before image). The logic in the Extract which matches inserts in the `DELETE TOMBSTONE` table to deletes also needs to be matched to PK updates, or unique key (UK) with at least one non-nullable field, if there is no PK.

Site priority needs support from the Replicat, both the parameters are implemented and the setting is passed to the apply.

Compatibility and Migration

Replicating from a base table which doesn't have a `KEYVER$$` to a target, which has `EARLIEST TIMESTAMP` resolution support, `DELETE ALWAYS WINS` resolution, or SITE PRIORITY, will receive an error in cases involving `DELETE` or PK Update.

Replicating from a base table, which has a `KEYVER$$` to a target, which does not, will ignore the `KEYVER$$`, when replicating to an earlier release, then the field is dropped or is not supported).

Example

The following table shows `EARLIEST TIMESTAMP` resolution on **Site1** with no `keyver$$` column or earlier RDBMS version replication to **Site 2**

Site 1	Site 2	Description
insert	insert	If the **Site1** `CDRTS$` timestamp is earlier then **Site1** wins else **Site2** wins
insert	update	Same as insert insert.
insert	delete	Conflict cannot be resolved, depending on configuration, error goes to error queue, discard and so on.
insert	pkupdate	Same as insert delete
update	insert	Same as insert insert
update	update	Same as insert insert
update	delete	Same as insert delete
update	pkupdate	Same as insert delete
delete	insert	Same as insert delete
delete	update	Same as insert delete
pkupdate	insert	Same as insert delete
pkupdate	update	Same as insert delete
pkupdate	delete	Same as insert delete

Primary Key Updates (pkupdate) interoperability will not resolve correctly without a backport. Besides the interoperability problems listed above, pkupdates that are replicated to earlier versions of the RDBMS, will not resolve correctly. A conflicting insert and replicated pkupdate on the earlier RDBMS may result in 2 rows. The insert will succeed to the original row and the pkupdate will succeed to update or create the new row.

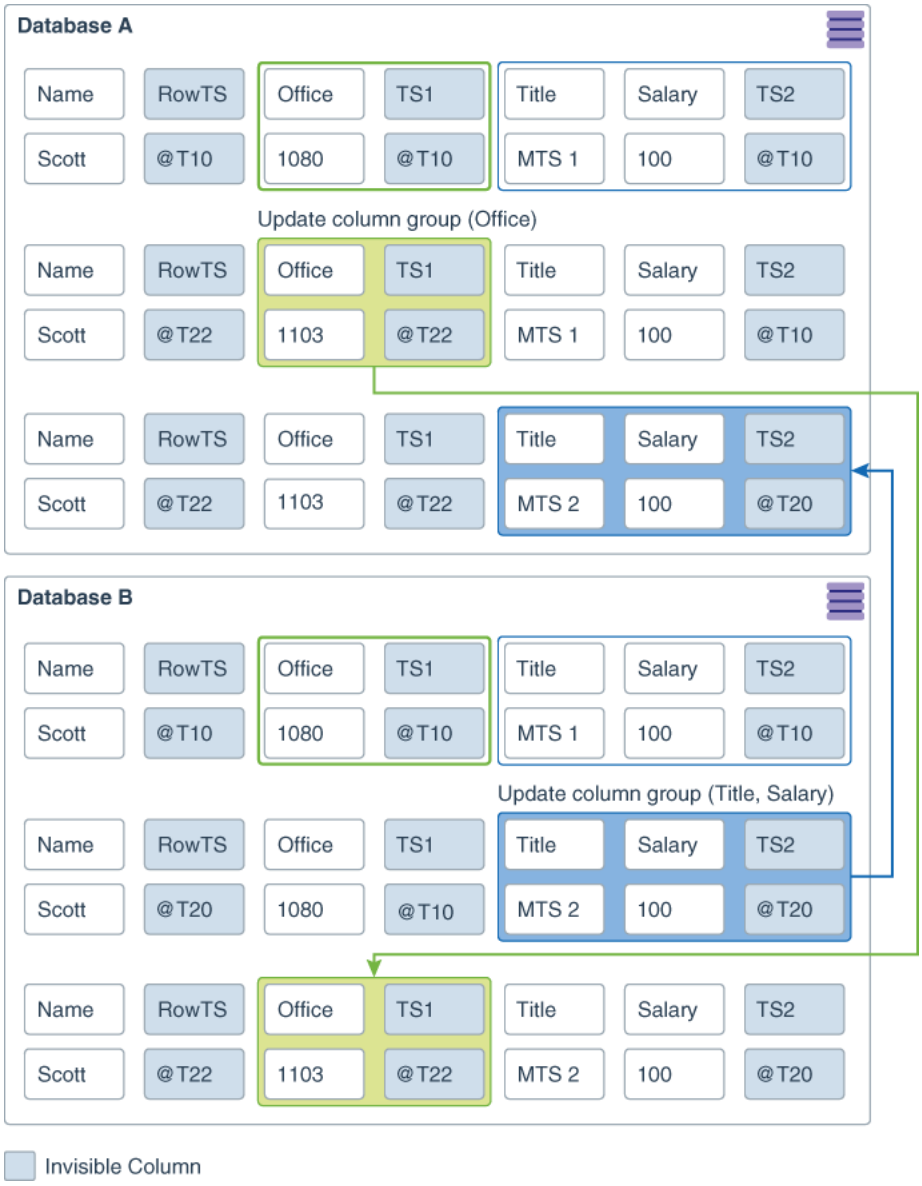
No upgrage or downgrade scripts are needed because the changes are just to procedure attributes and view columns.

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.



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 occurs

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 becasue some incoming updates and inserts may be delayed from another site and the incoming LCR needs to be filtered against the tombstone operation timestamp to determine whether it should be applied.

Earliest Timestamp Conflict Detection and Resolution

Columns with names of the form `CDRTS$ column group` and `CDRTS$ROW` are used to contain timestamps that reflect modification times for column groups and the row. The `DBMS_GOLDENGATE_ADM` procedures `ADD_AUTO_CDR()`, `ADD_AUTO_CDR_COLUMN_GROUP()`, `REMOVE_AUTO_CDR()`, `REMOVE_AUTO_CDR_COLUMN_GROUP()`, `ALTER_AUTO_CDR()`, and `ALTER_AUTO_CDR_COLUMN_GROUP()` are presently used to configure ACDR with latest timestamp resolution. They are also used in configuration of ACDR with earliest timestamp resolution, 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 `acdrflags_kqldtvc` to indicate 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

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 ignoreed, 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.

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.

Exmaple:

Key Version `kv` and Timestamp `ts`

Database 1: `insert tab1 key1 kv1 ts1`

Database 2: `delete tab1 key1 kv1 ts1`

Insertion to `DELETE TOMBSTONE` table `key1 kv1 ts1`

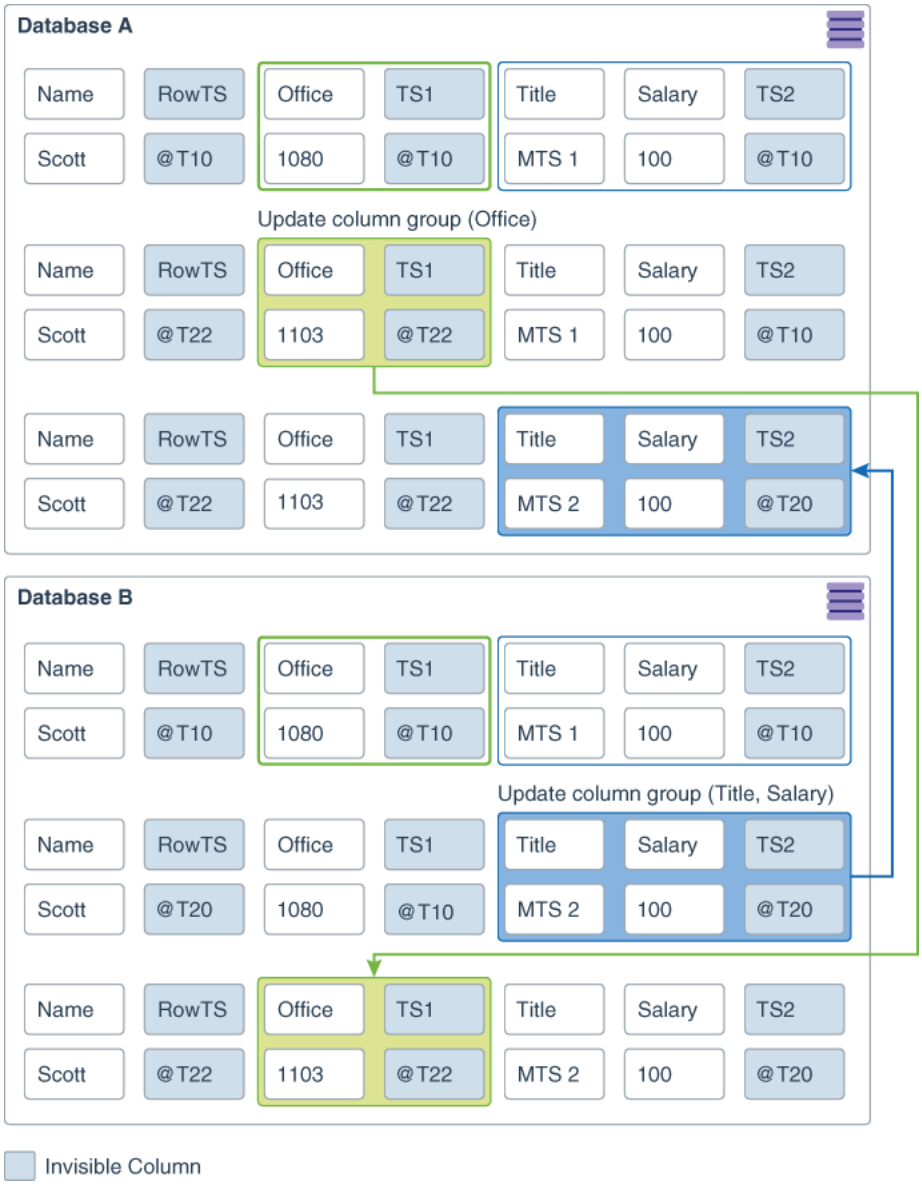
Database 1: `insert tab1 key1 kv2 ts2`

Without using the key version, the insert would be ignored, the delete always wins. As the key version is used, you know that `kv2` is not the version of the row that was deleted and the insert succeeds.

Delta Conflict Detection and 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.



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 `TS1` 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 `100` to `110`. Therefore, the value was increased by 10.
- At database B, the value of `Balance` was changed from `100` to `120`. Therefore, the value was increased by 20.
- 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 20 (120–100=20). Therefore, the current value in the table (110) is increased by 20 so that the value after conflict resolution is 130.

- 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 10 (110–100=10). Therefore, the current value in the table (120) is increased by 10 so that the value after conflict resolution is 130.

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 specified in Replicat parameter file between source and target in conflict resolution.

`SITE PRIORITY` is turned on for a database or PDB in the Replicat parameter file with the parameter `ACDR SITE_PRIORITY {source_db_name}{OVERWRITE | IGNORE }` specified to turn on `SITE PRIORITY` resolution for a table. If `OVERWRITE` is specified, then the source table has priority and conflicts are resolved by `OVERWRITE`. Conversely, if `IGNORE` is specified, then the target table has priority and source table change are ignored in a conflict. `SITE PRIORITY` resolution can be turned off by the field `ADDITIONAL_OPTIONS` in both `DBMS_GOLDENGATE_ADM` procedure `ADD_AUTO_CDR()` and `ALTER_AUTO_CDR()` setting the bit `IGNORE_SITE_PRIORITY`. Every Replicat source-target relationship can be set up differently, therefore, convergence is dependent on user setup.

Track PK 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 11-21 Configuring Delta Conflict Detection and Resolution for a Table

This example configures delta conflict detection and resolution for the `order_total` column in the `oe.orders` table.

```
BEGIN
  DBMS_GOLDENGATE_ADM.ADD_AUTO_CDR(
    SCHEMA_NAME => 'OE',
    TABLE_NAME  => 'ORDERS');
END;
/

BEGIN
  DBMS_GOLDENGATE_ADM.ADD_AUTO_CDR_DELTA_RES(
    SCHEMA_NAME => 'OE',
```

```
    TABLE_NAME  => 'ORDERS',
    COLUMN_NAME => 'ORDER_TOTAL');
END;
/
```

Configuring Latest Timestamp Conflict Detection and Resolution

The `ADD_AUTO_CDR` procedure in the `DBMS_GOLDENGATE_ADM` package configures latest timestamp conflict detection and resolution. The `ADD_AUTO_CDR_COLUMN_GROUP` procedure adds optional column groups.

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. Optional: 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 11-22 Configuring the Latest Timestamp Conflict Detection and Resolution for a Table

This example configures latest timestamp conflict detection and resolution for the `hr.employees` table.

```
BEGIN
  DBMS_GOLDENGATE_ADM.ADD_AUTO_CDR(
    schema_name => 'hr',
    table_name  => 'employees');
END;
/
```

Example 11-23 Configuring Column Groups

This example configures the following column groups for timestamp conflict resolution on the `hr.employees` table:

- The `job_identifier_cg` column group includes the `job_id`, `department_id`, and `manager_id` columns.
- The `compensation_cg` column group includes the `salary` and `commission_pct` columns.

```
BEGIN
  DBMS_GOLDENGATE_ADM.ADD_AUTO_CDR_COLUMN_GROUP(
```

```
    schema_name       => 'hr',
    table_name        => 'employees',
    column_list       => 'job_id,department_id,manager_id',
    column_group_name => 'job_identifier_cg');
END;
/

BEGIN
  DBMS_GOLDENGATE_ADM.ADD_AUTO_CDR_COLUMN_GROUP(
    schema_name       => 'hr',
    table_name        => 'employees',
    column_list       => 'salary,commission_pct',
    column_group_name => 'compensation_cg');
END;
/
```

Configuring Delta Conflict Detection and Resolution

The `ADD_AUTO_CDR_DELTA_RES` procedure in the `DBMS_GOLDENGATE_ADM` package configures delta conflict detection and resolution.

With delta conflict resolution, you specify one column for which conflicts are detected and resolved. The conflict is detected if the value of the column in the row LCR does not match the corresponding value in the table. The conflict is resolved by adding the difference between the new and old values in the row LCR to the value in the table.

You can configure an Oracle GoldenGate administrator using the `GRANT_ADMIN_PRIVILEGE` procedure in the `DBMS_GOLDENGATE_ADM` package.

1. Connect to the inbound server database as an Oracle GoldenGate administrator.
2. Run the `ADD_AUTO_CDR` procedure and specify the table to configure for latest timestamp conflict detection and resolution.
3. Run the `ADD_AUTO_CDR_DELTA_RES` procedure and specify the column on which delta conflict detection and resolution is performed.
4. Repeat the previous steps in each Oracle Database that replicates the table.

Example 11-24 Configuring Delta Conflict Detection and Resolution for a Table

This example configures delta conflict detection and resolution for the `order_total` column in the `oe.orders` table.

```
BEGIN
  DBMS_GOLDENGATE_ADM.ADD_AUTO_CDR(
    SCHEMA_NAME => 'OE',
    TABLE_NAME  => 'ORDERS');
END;
/

BEGIN
  DBMS_GOLDENGATE_ADM.ADD_AUTO_CDR_DELTA_RES(
    SCHEMA_NAME => 'OE',
    TABLE_NAME  => 'ORDERS',
    COLUMN_NAME => 'ORDER_TOTAL');
```

```
END;
/
```

Managing Automatic Conflict Detection and Resolution

You can manage Oracle GoldenGate automatic conflict detection and resolution in Oracle Database with the `DBMS_GOLDENGATE_ADM` package.

Altering Conflict Detection and Resolution for a Table

The `ALTER_AUTO_CDR` procedure in the `DBMS_GOLDENGATE_ADM` package alters conflict detection and resolution for a table.

Oracle GoldenGate automatic conflict detection and resolution must be configured for the table:

1. Connect to the inbound server database as the Oracle GoldenGate administrator.
2. Run the `ALTER_AUTO_CDR` procedure and specify the table to configure for latest timestamp conflict detection and resolution.
3. Repeat all of the previous steps in each Oracle Database that replicates the table.

Example 11-25 Altering Conflict Detection and Resolution for a Table

This example alters conflict detection and resolution for the `HR.EMPLOYEES` table to specify that delete conflicts are tracked in a tombstone table.

```
BEGIN
  DBMS_GOLDENGATE_ADM.ALTER_AUTO_CDR(
    SCHEMA_NAME       => 'HR',
    TABLE_NAME        => 'EMPLOYEES',
    TOMBSTONE_DELETES => TRUE);
END;
/
```

Altering a Column Group

The `ALTER_AUTO_CDR_COLUMN_GROUP` procedure alters a column group.

1. Connect to the inbound server database as an Oracle GoldenGate administrator.
2. Run the `ALTER_AUTO_CDR_COLUMN_GROUP` procedure and specify one or more column groups in the table.
3. Repeat all of the previous steps in each Oracle Database that replicates the table.

Example 11-26 Altering a Column Group

This example removes the `MANAGER_ID` column from the `JOB_IDENTIFIER_CG` column group for the `HR.EMPLOYEES` table.

```
BEGIN
  DBMS_GOLDENGATE_ADM.ALTER_AUTO_CDR_COLUMN_GROUP(
    SCHEMA_NAME        => 'HR',
    TABLE_NAME         => 'EMPLOYEES',
    COLUMN_GROUP_NAME  => 'JOB_IDENTIFIER_CG',
    REMOVE_COLUMN_LIST => 'MANAGER_ID');
```

```
END;
/
```

> **Note:**
>
> If there is more than one column, then use a comma-separated list.

Purging Tombstone Rows

The `PURGE_TOMBSTONES` procedure removes tombstone rows that were recorded before a specified date and time. This procedure removes the tombstone rows for all tables configured for conflict resolution in the database.

It might be necessary to purge tombstone rows periodically to keep the tombstone log from growing too large over time.

1. Connect to the inbound server database as an Oracle GoldenGate administrator.
2. Run the `PURGE_TOMBSTONES` procedure and specify the date and time.

Example 11-27 Purging Tombstone Rows

This example purges all tombstone rows recorded before 3:00 p.m. on December, 1, 2015 Eastern Standard Time. The timestamp must be entered in `TIMESTAMP WITH TIME ZONE` format.

```
EXEC DBMS_GOLDENGATE_ADM.PURGE_TOMBSTONES('2015-12-01 15:00:00.000000 EST');
```

Removing Conflict Detection and Resolution From a Table

The `REMOVE_AUTO_CDR` procedure in the `DBMS_GOLDENGATE_ADM` package removes automatic conflict detection and resolution from a table. This procedure also removes any column groups and delta conflict detection and resolution configured for the table.

1. Connect to the inbound server database as an Oracle GoldenGate administrator.
2. Run the `REMOVE_AUTO_CDR` procedure and specify the table.
3. Repeat all of the previous steps in each Oracle Database that replicates the table.

Example 11-28 Removing Conflict Detection and Resolution for a Table

This example removes conflict detection and resolution for the `hr.employees` table.

```
BEGIN
  DBMS_GOLDENGATE_ADM.REMOVE_AUTO_CDR(
    schema_name => 'hr',
    table_name  => 'employees');
END;
/
```

Removing a Column Group

The `REMOVE_AUTO_CDR_COLUMN_GROUP` procedure removes a column group.

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 11-29 Removing a Column Group

This example removes the `compensation_cg` column group from the `hr.employees` table.

```
BEGIN
  DBMS_GOLDENGATE_ADM.REMOVE_AUTO_CDR_COLUMN_GROUP(
    schema_name       => 'hr',
    table_name        => 'employees',
    column_group_name => 'compensation_cg');
END;
/
```

Removing Delta Conflict Detection and Resolution

The `REMOVE_AUTO_CDR_DELTA_RES` procedure in the `DBMS_GOLDENGATE_ADM` package removes delta conflict detection and resolution for a column.

Delta conflict detection and resolution must be configured for the specified column.

1. Connect to the inbound server database as an Oracle GoldenGate administrator.
2. Run the `REMOVE_AUTO_CDR_DELTA_RES` procedure and specify the column.
3. Repeat all of the previous steps in each Oracle Database that replicates the table.

Example 11-30 Removing Delta Conflict Detection and Resolution for a Table

This example removes delta conflict detection and resolution for the `ORDER_TOTAL` column in the `OE.ORDERS` table.

```
BEGIN
  DBMS_GOLDENGATE_ADM.REMOVE_AUTO_CDR_DELTA_RES(
    SCHEMA_NAME => 'OE',
    TABLE_NAME  => 'ORDERS',
    COLUMN_NAME => 'ORDER_TOTAL');
END;
/
```

Monitoring Automatic Conflict Detection and Resolution

You can monitor Oracle GoldenGate automatic conflict detection and resolution in an Oracle Database by querying data dictionary views.

Displaying Information About the Tables Configured for Conflicts

The `ALL_GG_AUTO_CDR_TABLES` view displays information about the tables configured for Oracle GoldenGate automatic conflict detection and resolution.

1. Connect to the database.
2. Query the `ALL_GG_AUTO_CDR_TABLES` view.

Example 11-31 Displaying Information About the Tables Configured for Conflict Detection and Resolution

This query displays the following information about the tables that are configured for conflict detection and resolution:

- The table owner for each table.
- The table name for each table.
- The tombstone table used to store rows deleted for update-delete conflicts, if a tombstone table is configured for the table.
- The hidden timestamp column used for conflict resolution for each table.

```
COLUMN TABLE_OWNER FORMAT A15
COLUMN TABLE_NAME FORMAT A15
COLUMN TOMBSTONE_TABLE FORMAT A15
COLUMN ROW_RESOLUTION_COLUMN FORMAT A25

SELECT TABLE_OWNER,
       TABLE_NAME,
       TOMBSTONE_TABLE,
       ROW_RESOLUTION_COLUMN
  FROM ALL_GG_AUTO_CDR_TABLES
  ORDER BY TABLE_OWNER, TABLE_NAME;
```

Your output looks similar to the following:

```
TABLE_OWNER     TABLE_NAME      TOMBSTONE_TABLE ROW_RESOLUTION_COLUMN
--------------- --------------- --------------- -------------------------
HR              EMPLOYEES       DT$_EMPLOYEES   CDRTS$ROW
OE              ORDERS          DT$_ORDERS      CDRTS$ROW
```

Displaying Information About Conflict Resolution Columns

The `ALL_GG_AUTO_CDR_COLUMNS` view displays information about the columns configured for Oracle GoldenGate automatic conflict detection and resolution.

The columns can be configured for row or column automatic conflict detection and resolution. The columns can be configured for latest timestamp conflict resolution in a column group. In addition, a column can be configured for delta conflict resolution.

1. Connect to the database as an Oracle GoldenGate administrator.
2. Query the `ALL_GG_AUTO_CDR_COLUMNS` view.

Example 11-32 Displaying Information About Column Groups

This query displays the following information about the tables that are configured for conflict detection and resolution:

- The table owner for each table.
- The table name for each table.
- If the column is in a column group, then the name of the column group.
- The column name.

- If the column is configured for latest timestamp conflict resolution, then the name of the hidden timestamp column for the column.

```
COLUMN TABLE_OWNER FORMAT A10
COLUMN TABLE_NAME FORMAT A10
COLUMN COLUMN_GROUP_NAME FORMAT A17
COLUMN COLUMN_NAME FORMAT A15
COLUMN RESOLUTION_COLUMN FORMAT A23

SELECT TABLE_OWNER,
       TABLE_NAME,
       COLUMN_GROUP_NAME,
       COLUMN_NAME,
       RESOLUTION_COLUMN
  FROM ALL_GG_AUTO_CDR_COLUMNS
  ORDER BY TABLE_OWNER, TABLE_NAME;
```

Your output looks similar to the following:

```
TABLE_OWNE TABLE_NAME COLUMN_GROUP_NAME COLUMN_NAME     RESOLUTION_COLUMN
---------- ---------- ----------------- ---------------
-----------------------
HR         EMPLOYEES  COMPENSATION_CG   COMMISSION_PCT  CDRTS$COMPENSATION_CG
HR         EMPLOYEES  COMPENSATION_CG   SALARY          CDRTS$COMPENSATION_CG
HR         EMPLOYEES  JOB_IDENTIFIER_CG MANAGER_ID
CDRTS$JOB_IDENTIFIER_CG
HR         EMPLOYEES  JOB_IDENTIFIER_CG JOB_ID
CDRTS$JOB_IDENTIFIER_CG
HR         EMPLOYEES  JOB_IDENTIFIER_CG DEPARTMENT_ID
CDRTS$JOB_IDENTIFIER_CG
HR         EMPLOYEES  IMPLICIT_COLUMNS$ PHONE_NUMBER    CDRTS$ROW
HR         EMPLOYEES  IMPLICIT_COLUMNS$ LAST_NAME       CDRTS$ROW
HR         EMPLOYEES  IMPLICIT_COLUMNS$ HIRE_DATE       CDRTS$ROW
HR         EMPLOYEES  IMPLICIT_COLUMNS$ FIRST_NAME      CDRTS$ROW
HR         EMPLOYEES  IMPLICIT_COLUMNS$ EMAIL           CDRTS$ROW
HR         EMPLOYEES  IMPLICIT_COLUMNS$ EMPLOYEE_ID     CDRTS$ROW
OE         ORDERS     IMPLICIT_COLUMNS$ ORDER_MODE      CDRTS$ROW
OE         ORDERS     IMPLICIT_COLUMNS$ ORDER_ID        CDRTS$ROW
OE         ORDERS     IMPLICIT_COLUMNS$ ORDER_DATE      CDRTS$ROW
OE         ORDERS     IMPLICIT_COLUMNS$ CUSTOMER_ID     CDRTS$ROW
OE         ORDERS     DELTA$            ORDER_TOTAL
OE         ORDERS     IMPLICIT_COLUMNS$ PROMOTION_ID    CDRTS$ROW
OE         ORDERS     IMPLICIT_COLUMNS$ ORDER_STATUS    CDRTS$ROW
OE         ORDERS     IMPLICIT_COLUMNS$ SALES_REP_ID    CDRTS$ROW
```

In this example, the columns with `IMPLICIT_COLUMNS$` for the column group name are configured for row conflict detection and resolution, but they are not part of a column group. The columns with `DELTA$` for the column group name are configured for delta conflict detection and resolution, and these columns do not have a resolution column.

Displaying Information About Column Groups

The `ALL_GG_AUTO_CDR_COLUMN_GROUPS` view displays information about the column groups configured for Oracle GoldenGate automatic conflict detection and resolution.

You can configure Oracle GoldenGate automatic conflict detection and resolution using the `ADD_AUTO_CDR` procedure in the `DBMS_GOLDENGATE_ADM` package. You can configure column groups using the `ADD_AUTO_CDR_COLUMN_GROUP` procedure in the `DBMS_GOLDENGATE_ADM` package.

1. Connect to the database as an Oracle GoldenGate administrator.
2. Query the `ALL_GG_AUTO_CDR_COLUMN_GROUPS` view.

Example 11-33 Displaying Information About Column Groups

This query displays the following information about the tables that are configured for conflict detection and resolution:

- The table owner.
- The table name.
- The name of the column group.
- The hidden timestamp column used for conflict resolution for each column group.

```
COLUMN TABLE_OWNER FORMAT A15
COLUMN TABLE_NAME FORMAT A15
COLUMN COLUMN_GROUP_NAME FORMAT A20
COLUMN RESOLUTION_COLUMN FORMAT A25

SELECT TABLE_OWNER,
       TABLE_NAME,
       COLUMN_GROUP_NAME,
       RESOLUTION_COLUMN
  FROM ALL_GG_AUTO_CDR_COLUMN_GROUPS
  ORDER BY TABLE_OWNER, TABLE_NAME;
```

The output looks similar to the following:

```
TABLE_OWNER     TABLE_NAME      COLUMN_GROUP_NAME    RESOLUTION_COLUMN
--------------- --------------- -------------------- ------------------------
HR              EMPLOYEES       COMPENSATION_CG      CDRTS$COMPENSATION_CG
HR              EMPLOYEES       JOB_IDENTIFIER_CG    CDRTS$JOB_IDENTIFIER_CG
```

Manual Conflict Detection and Resolution

Learn about manually configuring Conflict Detection and Resolution (CDR) using specific parameters. Conflict detection and resolution is required in active-active configurations, where Oracle GoldenGate must maintain data synchronization among multiple databases that contain the same data sets.

Overview of the Oracle GoldenGate CDR Feature

Oracle GoldenGate Conflict Detection and Resolution (CDR) 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
- 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. Do not use CDR for columns that contain LOBs, abstract data types (ADT), or user-defined types (UDT).

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. For more information, see *Parameters and Functions Reference for Oracle GoldenGate*.

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: HANDLECOLLSIONS, 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 statment, 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 `INSERT`s 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.

```
    -- REPERROR error handling: DEFAULT represents all error types. DISCARD
    -- writes operations that could not be processed to a discard file.
REPERROR (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.
dberrmsge = @GETENV ('LASTERR', 'DBERRMSG'), &
    -- captures and maps the name of the target table
tabname = @GETENV ('GGHEADER', 'TABLENAME'), &
    -- If the names and definitions of the source columns and the target
    -- exceptions columns were not identical, the columns would need to
    -- be mapped in the COLMAP clause instead of using USEDEFAULTS, as
    -- follows:
       -- name_after = name, &
       -- phone_after = phone, &
       -- address_after = address, &
       -- salary_after = salary, &
       -- balance_after = balance, &
       -- comment_after = comment, &
       -- last_mod_time_after = last_mod_time &
    -- maps the before image of each column from the trail to a column in the
    -- exceptions table.
name_before = @BEFORE (name), &
phone_before = @BEFORE (phone), &
address_before = @BEFORE (address), &
salary_before = @BEFORE (salary), &
balance_before = @BEFORE (balance), &
comment_before = @BEFORE (comment), &
last_mod_time_before = @BEFORE (last_mod_time), &
    -- maps the results of the SQLEXEC query to rows in the exceptions table
    -- to show the current image of the row in the target.
name_current = qry.name, &
phone_current = qry.phone, &
address_current = qry.address, &
salary_current = qry.salary, &
balance_current = qry.balance, &
comment_current = qry.comment, &
last_mod_time_current = qry.last_mod_time)
;
```

Once you are confident that your routines work as expected in all situations, you can reduce the amount of data that is logged to the exceptions table to reduce the overhead of the resolution routines.

Configuring the Oracle GoldenGate Parameter Files for Conflict Resolution

The following parameters are required to support conflict detection and resolution.

1. Use the `COMPARECOLS` option of the `MAP` parameter in the Replicat parameter file to specify columns that are to be used with before values in the Replicat `WHERE` clause. The before values are compared with the current values in the target database to detect update and delete conflicts. (By default, Replicat only uses the primary key in the `WHERE` clause; this may not be enough for conflict detection).

2. Use the `RESOLVECONFLICT` option of the `MAP` parameter to specify conflict resolution routines for different operations and conflict types. You can use `RESOLVECONFLICT` multiple times in a `MAP` statement to specify different resolutions for different conflict types. However, you cannot use `RESOLVECONFLICT` multiple times for the same type of conflict. Use identical conflict-resolution procedures on all databases, so that the same conflict produces the same end result. One conflict-resolution method might not work for every conflict that could occur. You might need to create several routines that can be called in a logical order of priority so that the risk of failure is minimized.

> **Note:**
>
> Additional consideration should be given when a table has a primary key and additional unique indexes or unique keys. The automated routines provided with the `COMPARECOLS` and `RESOLVECONFLICT` parameters require a consistent way to uniquely identify each row. Failure to consistently identify a row will result in an error during conflict resolution. In these situations the additional unique keys should be disabled or you can use the `SQLEXEC` feature to handle the error thrown and resolve the conflict.

For detailed information about these parameters, see *Parameters and Functions Reference for Oracle GoldenGate*. See the examples starting on CDR Example 1: All Conflict Types with USEMAX, OVERWRITE, DISCARD, for more information on these parameters.

Making the Required Column Values Available to Extract

To use CDR, the following column values must be logged so that Extract can write them to the trail.

- The full before image of each record. Some databases do not provide a before image in the log record, and must be configured to do so with supplemental logging. For most supported databases, you can use the `ADD TRANDATA` command for this purpose.
- Use the `LOGALLSUPCOLS` parameter to ensure that the full before and after images of the scheduling columns are written to the trail. Scheduling columns are primary key, unique index, and foreign key columns. `LOGALLSUPCOLS` causes Extract to include in the trail record the before image for `UPDATE` operations and the before image of all supplementally logged columns for both `UPDATE` and `DELETE` operations.

For detailed information about these parameters and commands, see the *Parameters and Functions Reference for Oracle GoldenGate*. See the examples starting on CDR Example 1: All Conflict Types with USEMAX, OVERWRITE, DISCARD for more information on how these parameters work with CDR.

Viewing CDR Statistics

The CDR feature provides the following methods for viewing the results of conflict resolution.

Here are different techniques you can use to view CDR statistics.

Report File

Replicat writes CDR statistics to the report file:

```
Total CDR conflicts        7
    CDR resolutions succeeded       6
    CDR resolutions failed          1
    CDR INSERTROWEXISTS conflicts   1
    CDR UPDATEROWEXISTS conflicts   4
    CDR UPDATEROWMISSING conflicts
    CDR DELETEROWEXISTS conflicts      1
    CDR DELETEROWMISSING conflicts     1
```

Command Line

You can view CDR statistics from the command line by using the `STATS REPLICAT` command with the `REPORTCDR` option:

```
STATS REPLICAT group, REPORTCDR
```

Column-conversion Functions

The following CDR statistics can be retrieved and mapped to an exceptions table or used in other Oracle GoldenGate parameters that accept input from column-conversion functions, as appropriate.

- Number of conflicts that Replicat detected
- Number of resolutions that the Replicat resolved
- Number of resolutions that the Replicat could not resolve

To retrieve these statistics, use the `@GETENV` column-conversion function with the `STATS` or `DELTASTATS` information type. The results are based on the current Replicat session. If Replicat stops and restarts, it resets the statistics.

You can return these statistics for a specific table or set of wildcarded tables:

```
@GETENV ('STATS','TABLE','SCHEMA.TABLNAME','CDR_CONFLICTS')
@GETENV ('STATS','TABLE','SCHEMA.TABLNAME','CDR_RESOLUTIONS_SUCCEEDED')
@GETENV ('STATS','TABLE','SCHEMA.TABLNAME','CDR_RESOLUTIONS_FAILED')
```

You can return these statistics for all of the tables in all of the `MAP` statements in the Replicat parameter file:

```
@GETENV ('STATS','CDR_CONFLICTS')
@GETENV ('STATS','CDR_RESOLUTIONS_SUCCEEDED')
@GETENV ('STATS','CDR_RESOLUTIONS_FAILED')
```

The `'STATS'` information type in the preceding examples can be replaced by `'DELTASTATS'` to return the requested counts since the last execution of `'DELTASTATS'`. For more information about `@GETENV`, see @GETENV

CDR Example 1: All Conflict Types with USEMAX, OVERWRITE, DISCARD

This example resolves all conflict types by using the `USEMAX`, `OVERWRITE`, and `DISCARD` resolutions.

Table Used in this Example

The examples assume identical Oracle databases.

```
CREATE TABLE 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 with Conflict Resolution Specifications

```
MAP fin.src, TARGET fin.tgt,
    COMPARECOLS (ON UPDATE ALL, ON DELETE ALL),
    RESOLVECONFLICT (UPDATEROWEXISTS, (DEFAULT, USEMAX (last_mod_time)),
    RESOLVECONFLICT (INSERTROWEXISTS, (DEFAULT, USEMAX (last_mod_time)),
    RESOLVECONFLICT (DELETEROWEXISTS, (DEFAULT, OVERWRITE)),
    RESOLVECONFLICT (UPDATEROWMISSING, (DEFAULT, OVERWRITE)),
    RESOLVECONFLICT (DELETEROWMISSING, (DEFAULT, DISCARD)),
    );
```

Description of MAP Statement

The following describes the `MAP` statement:

- Per `COMPARECOLS`, use the before image of all columns in the trail record in the Replicat `WHERE` clause for updates and deletes.
- Per `DEFAULT`, use all columns as the column group for all conflict types; thus the resolution applies to all columns.
- For an `INSERTROWEXISTS` conflict, use the `USEMAX` resolution: If the row exists during an insert, use the `last_mod_time` column as the resolution column for deciding which is the greater value: the value in the trail or the one in the database. If the value in the trail is greater, apply the record but change the insert to an update. If the database value is higher, ignore the record.
- For an `UPDATEROWEXISTS` conflict, use the `USEMAX` resolution: If the row exists during an update, use the `last_mod_time` column as the resolution column: If the value in the trail is greater, apply the update.
- If you use `USEMIN` or `USEMAX`, and the values are exactly the same, then `RESOLVECONFLICT` isn't triggered and the incoming row is ignored. If you use `USEMINEQ` or `USEMAXEQ`, and the values are exactly the same, then the resolution is triggered.

- For a DELETEROWEXISTS conflict, use the OVERWRITE resolution: If the row exists during a delete operation, apply the delete.
- For an UPDATEROWMISSING conflict, use the OVERWRITE resolution: If the row does not exist during an update, change the update to an insert and apply it.
- For a DELETROWMISSING conflict use the DISCARD resolution: If the row does not exist during a delete operation, discard the trail record.

> **Note:**
>
> As an alternative to USEMAX, you can use the USEMAXEQ resolution to apply a >= condition. For more information, see *Parameters and Functions Reference for Oracle GoldenGate*.

INSERTROWEXISTS with the USEMAX Resolution

For this example, the USEMAX resolution is illustrated with the applicable before and after images for the record in the trail and in the database. It shows how to resolve an insert where the row exists in the source and target, but some or all row values are different.

Table 11-10 INSERTROWEXISTS Conflict with USEMAX Resolution

Image	SQL	Comments
Before image in trail	None (row was inserted on the source).	N/A
After image in trail	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 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='111111' address='Ralston' salary=200 balance=500 comment='aaa' last_mod_time='9/1/10 1:00'	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: 1)'Mary' 2)'1234567890' 3)'Oracle Pkwy' 4)100 5)100 6)NULL 7)'9/1/10 3:00'	This SQL returns a uniqueness conflict on 'Mary'.

Table 11-10 (Cont.) INSERTROWEXISTS Conflict with USEMAX Resolution

Image	SQL	Comments
UPDATE applied by Replicat to resolve the conflict	SQL bind variables: 1)'1234567890' 2)'Oracle Pkwy' 3)100 4)100 5)NULL 6)'9/1/10 3:00' 7)'Mary' 8)'9/1/10 3:00'	Because USEMAX is specified for INSERTROWEXISTS, Replicat converts the insert to an update, and it compares the value of last_mod_time in the trail record with the value in the database. The value in the record is greater, so the after images for columns in the trail file are applied to the target.

UPDATEROWEXISTS with the USEMAX Resolution

For this example, the USEMAX resolution is illustrated with the applicable before and after images for the record in the trail and in the database. It shows how to resolve an update where the row exists in the source and target, but some or all row values are different.

Table 11-11 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.

Table 11-11 (Cont.) UPDATEROWEXISTS Conflict with USEMAX Resolution

Image	SQL	Comments

Initial `UPDATE` applied by Replicat that detects the conflict	SQL bind variables:
	`1)'222222'`
	`2)'Holly'`
	`3)'9/1/10 5:00'`
	`4)'Mary'`
	`5)'1234567890'`
	`6)'Oracle Pkwy'`
	`7)100`
	`8)100`
	`9)NULL`
	`10)'9/1/10 3:00'`
		This SQL returns a no-data-found error because the values for the `balance`, `comment`, and `last_mod_time` are different in the target.
		All columns are used in the `WHERE` clause because the `COMPARECOLS` statement is set to `ALL`.

`UPDATE` applied by Replicat to resolve the conflict	SQL bind variables:
	`1)'Mary'`
	`2)'222222'`
	`3)'Holly'`
	`4)100`
	`5)100`
	`6)NULL`
	`7)'9/1/10 5:00'`
	`8)'Mary'`
	`9)'9/1/10 5:00'`
		Because the after value of `last_mod_time` in the trail record is less than the current value in the database, the database value is retained. Replicat applies the operation with a `WHERE` clause that contains the primary key plus a `last_mod_time` value set to less than `9/1/10 5:00`. No rows match this criteria, so the statement fails with a "data not found" error, but Replicat ignores the error because a `USEMAX` resolution is expected to fail if the condition is not satisfied.

UPDATEROWMISSING with OVERWRITE Resolution

For this example, the `OVERWRITE` resolution is illustrated with the applicable before and after images for the record in the trail and in the database. It shows how to resolve the case where the target row is missing. The logical resolution, and the one used, is to overwrite the row into the target so that both databases are in sync again.

Table 11-12 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)`	

Table 11-12 (Cont.) UPDATEROWMISSING Conflict with OVERWRITE Resolution

Image	SQL	Comments

Initial `UPDATE` applied by Replicat that detects the conflict	SQL bind variables:
	`1)'4444'`
	`2)'Holly'`
	`3)'9/1/10 8:00'`
	`4)'Jane'`
	`5)'333'`
	`6)'Oracle Pkwy'`
	`7)200`
	`8)200`
	`9)NULL`
	`10)'9/1/10 7:00'`
		This SQL returns a no-data-found error. All columns are used in the `WHERE` clause because the `COMPARECOLS` statement is set to `ALL`.

`INSERT` applied by Replicat to resolve the conflict	SQL bind variables:
	`1)'Jane'`
	`2)'4444'`
	`3)'Holly'`
	`4)200`
	`5)200`
	`6)NULL`
	`7)'9/1/10 8:00'`
		The update is converted to an insert because `OVERWRITE` is the resolution. The after image of a column is used if available; otherwise the before image is used.

DELETEROWEXISTS with OVERWRITE Resolution

For this example, the `OVERWRITE` resolution is illustrated with the applicable before and after images for the record in the trail and in the database. It shows how to resolve the case where the source row was deleted but the target row exists. In this case, the `OVERWRITE` resolution applies the delete to the target.

Table 11-13 DELETEROWEXISTS Conflict with OVERWRITE Resolution

Image	SQL	Comments

Before image in trail	`name='Mary'`
	`phone='222222'`
	`address='Holly'`
	`salary=100`
	`balance=100`
	`comment=NULL`
	`last_mod_time='9/1/10 5:00'`
		N/A

After image in trail	`None`	N/A

Target database image	`name='Mary'`
	`phone='1234567890'`
	`address='Oracle Pkwy'`
	`salary=100`
	`balance=600`
	`comment=com`
	`last_mod_time='9/1/10 7:00'`
		The row exists on the target, but the `phone`, `address`, `balance`, `comment`, and `last_mod_time` columns are different from the before image in the trail.

Table 11-13 (Cont.) DELETEROWEXISTS Conflict with OVERWRITE Resolution

Image	SQL	Comments
Initial DELETE applied by Replicat that detects the conflict	SQL bind variables: 1)'Mary' 2)'222222' 3)'Holly' 4)100 5)100d 6)NULL 7)'9/1/10 5:00'	All columns are used in the WHERE clause because the COMPARECOLS statement is set to ALL. A no-data-found error occurs because of the difference between the before and current values.
DELETE applied by Replicat to resolve the conflict	SQL bind variables: 1)'Mary'	Because OVERWRITE is the resolution. the DELETE is applied using only the primary key (to avoid an integrity error).

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 11-14 DELETEROWMSING Conflict with DISCARD Resolution

Image	SQL	Comments
Before image in trail	name='Jane' phone='4444' address='Holly' salary=200 balance=200 comment=NULL last_mod_time='9/1/10 8:00'	N/A
After image in trail	None	N/A
Target database image	None (row missing)	N/A
Initial DELETE applied by Replicat that detects the conflict	SQL bind variables: 1)'Jane' 2)'4444' 3)'Holly' 4)200 5)200 6)NULL 7)'9/1/10 8:00'	This SQL returns a no-data-found error. All columns are used in the WHERE clause because the COMPARECOLS statement is set to ALL.
SQL applied by Replicat to resolve the conflict	None	Because DISCARD is specified as the resolution for 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 11-15 UPDATEROWEXISTS with USEDELTA and USEMAX

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` is the before image 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.

Target database image	`name='Mary'`
	`phone='1234567890'`
	`address='Oracle Pkwy'`
	`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.
		`salary=600` is the current image of the target column for the `USEDELTA` resolution.

Table 11-15 (Cont.) UPDATEROWEXISTS with USEDELTA and USEMAX

Image	SQL	Comments

Initial `UPDATE` applied by Replicat that detects the conflict	SQL bind variables:
	
	`1)'222222'`
	`2)'Holly'`
	`3)200`
	`4)'new'`
	`5)'9/1/10 5:00'`
	`6)'Mary'`
	`7)'1234567890'`
	`8)'Oracle Pkwy'`
	`9)100`
	`10)'9/1/10 3:00'`
		This SQL returns a no-data-found error because the values for the `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:
	
	`1)200`
	`2)100`
	`3)'Mary'`
		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:
	
	`1)'222222'`
	`2)'Holly'`
	`3)'new'`
	`4)'9/1/10 5:00'`
	`5)'Mary'`
	`6)'9/1/10 5:00'`
		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 11-16 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.
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.

Table 11-16 (Cont.) UPDATEROWEXISTS with USEDELTA, USEMAX, and IGNORE

Image	SQL	Comments
UPDATE applied by Replicat to resolve the conflict for USEMAX.	SQL bind variables: 1)'Holly' 2)'9/1/10 5:00' 3)'Mary' 4)'9/1/10 5:00'	Because the after value of 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.

Handling Processing Errors

This topic describes how to configure the Oracle GoldenGate processes to handle errors.

Oracle GoldenGate reports processing errors in several ways by means of its monitoring and reporting tools. For more information about these tools, see Monitoring Oracle GoldenGate Processing.

Topics:

Overview of Oracle GoldenGate Error Handling

Oracle GoldenGate provides error-handling options for:

- Extract
- Replicat
- TCP/IP

Handling Extract Errors

There is no specific parameter to handle Extract errors when DML operations are being extracted, but Extract does provide a number of parameters that can be used to prevent anticipated problems. These parameters handle anomalies that can occur during the processing of DML operations, such as what to do when a row to be fetched cannot be located, or what to do when the transaction log is not available. The following is a partial list of these parameters.

- FETCHOPTIONS
- WARNLONGTRANS
- DBOPTIONS
- TRANLOGOPTIONS

To handle extraction errors that relate to DDL operations, use the DDLERROR parameter.

For a complete parameter list, see *Parameters and Functions Reference for Oracle GoldenGate*.

Handling Replicat Errors during DML Operations

To control the way that Replicat responds to an error during one of its DML statements, use the `REPERROR` parameter in the Replicat parameter file. You can use `REPERROR` as a global parameter or as part of a `MAP` statement. You can handle most errors in a default fashion (for example, to cease processing) with `DEFAULT` and `DEFAULT2` options, and also handle other errors in a specific manner.

The following comprise the range of `REPERROR` responses:

- `ABEND`: roll back the transaction and stop processing.
- `DISCARD`: log the error to the discard file and continue processing.
- `EXCEPTION`: send the error for exceptions processing.
- `IGNORE`: ignore the error and continue processing.
- `RETRYOP [MAXRETRIES n]`: retry the operation, optionally up to a specific number of times.
- `TRANSABORT [, MAXRETRIES n] [, DELAY[C]SECS n]`: abort the transaction and reposition to the beginning, optionally up to a specific number of times at specific intervals.
- `RESET`: remove all previous `REPERROR` rules and restore the default of `ABEND`.
- `TRANSDISCARD`: discard the entire replicated source transaction if any operation within that transaction, including the commit, causes a Replicat error that is listed in the error specification. This option is useful when integrity constraint checking is disabled on the target.
- `TRANSEXCEPTION`: perform exceptions mapping for every record in the replicated source transaction, according to its exceptions-mapping statement, if any operation within that transaction (including the commit) causes a Replicat error that is listed in the error specification.

Most options operate on the individual record that generated an error, and Replicat processes the other, successful operations in the transaction. The exceptions are `TRANSDISCARD` and `TRANSEXCEPTION`: These options affect all records in a transaction if any record in that transaction generates an error. (The `ABEND` option also applies to the entire transaction, but does not apply error handling.)

See `REPERROR` for syntax and usage.

Handling Errors as Exceptions

When the action of `REPERROR` is `EXCEPTION` or `TRANSEXCEPTION`, you can map the values of operations that generate errors to an exceptions table and, optionally, map other information about the error that can be used to resolve the error. See About the Exceptions Table.

To map the exceptions to the exceptions table, use either of the following options of the `MAP` parameter:

- `MAP` with `EXCEPTIONSONLY`
- `MAP` with `MAPEXCEPTION`

Topics:

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 11-34 EXCEPTIONSONLY

This example shows how to use REPERROR with EXCEPTIONSONLY and an exceptions MAP statement. This example only shows the parameters that relate to REPERROR; other parameters not related to error handling are also required for Replicat.

```
REPERROR (DEFAULT, EXCEPTION)
MAP ggs.equip_account, TARGET ggs.equip_account2,
COLMAP (USEDEFAULTS);
MAP ggs.equip_account, TARGET ggs.equip_account_exception,
EXCEPTIONSONLY,
INSERTALLRECORDS
COLMAP (USEDEFAULTS,
DML_DATE = @DATENOW (),
OPTYPE = @GETENV ('LASTERR', 'OPTYPE'),
DBERRNUM = @GETENV ('LASTERR', 'DBERRNUM'),
DBERRMSG = @GETENV ('LASTERR', 'DBERRMSG'));
```

In this example, the REPERROR parameter is set for DEFAULT error handling, and the EXCEPTION option causes the Replicat process to treat failed operations as exceptions and continue processing.

Using MAPEXCEPTION

MAPEXCEPTION is valid when the names of the source and target tables in the MAP statement are wildcarded. Place the MAPEXCEPTION clause in the regular MAP statement, the same one where you map the source tables to the target tables. Replicat maps all operations that generate errors from all of the wildcarded tables to the same exceptions table; therefore, the exceptions table should contain a superset of all of the columns in all of the wildcarded tables.

Because you cannot individually map columns in a wildcard configuration, use the COLMAP clause with the USEDEFAULTS option to handle the column mapping for the wildcarded tables (or use the COLMATCH parameter if appropriate), and use explicit column mappings to map any additional information, such as that captured with column-conversion functions or SQLEXEC.

When using MAPEXCEPTION, include the INSERTALLRECORDS parameter in the MAPEXCEPTION clause. INSERTALLRECORDS causes all operation types to be applied to the exceptions table as INSERT operations. This is required to keep an accurate record of the exceptions and to prevent integrity errors on the exceptions table.

For more information about these parameters, see *Parameters and Functions Reference for Oracle GoldenGate*.

Example 11-35 MAPEXCEPTION

This is an example of how to use MAPEXCEPTION for exceptions mapping. The MAP and TARGET clauses contain wildcarded source and target table names. Exceptions that occur when processing any table with a name beginning with TRX are captured to the fin.trxexceptions table using the designated mapping.

```
MAP src.trx*, TARGET trg.*,
MAPEXCEPTION (TARGET fin.trxexceptions,
INSERTALLRECORDS,
COLMAP (USEDEFAULTS,
ACCT_NO = ACCT_NO,
OPTYPE = @GETENV ('LASTERR', 'OPTYPE'),
DBERR = @GETENV ('LASTERR', 'DBERRNUM'),
DBERRMSG = @GETENV ('LASTERR', 'DBERRMSG')
)
);
```

About the Exceptions Table

Use an exceptions table to capture information about an error that can be used for such purposes as troubleshooting your applications or configuring them to handle the error. At minimum, an exceptions table should contain enough columns to receive the entire row image from the failed operation. You can define extra columns to contain other information that is captured by means of column-conversion functions, `SQLEXEC`, or other external means.

To ensure that the trail record contains values for all of the columns that you map to the exceptions table, you can use either the `LOGALLSUPCOLS` parameter or the following parameters in the Extract parameter file:

- Use the `NOCOMPRESSDELETES` parameter so that all columns of a row are written to the trail for `DELETE` operations.
- Use the `GETUPDATEBEFORES` parameter so that Extract captures the before image of a row and writes them to the trail.

Handling Replicat errors during DDL Operations

To control the way that Replicat responds to an error that occurs for a DDL operation on the target, use the `DDLERROR` parameter in the Replicat parameter file. For more information, see *Parameters and Functions Reference for Oracle GoldenGate*.

Handling TCP/IP Errors

To provide instructions for responding to TCP/IP errors, use the `TCPERRS` file. This file is in the Oracle GoldenGate directory

Table 11-17 TCPERRS Columns

Column	Description
Error	Specifies a TCP/IP error for which you are defining a response.
Response	Controls whether or not Oracle GoldenGate tries to connect again after the defined error. Valid values are either `RETRY` or `ABEND`.
Delay	Controls how long Oracle GoldenGate waits before attempting to connect again.
Max Retries	Controls the number of times that Oracle GoldenGate attempts to connect again before aborting.

If a response is not explicitly defined in the `TCPERRS` file, Oracle GoldenGate responds to TCP/IP errors by abending.

Example 11-36 TCPERRS File

```
# TCP/IP error handling parameters
# Default error response is abend
#
# Error          Response   Delay(csecs)  Max Retries

ECONNABORTED     RETRY      1000          10
ECONNREFUSED     RETRY      1000          12
ECONNRESET       RETRY      500           10
ENETDOWN         RETRY      3000          50
```

```
ENETRESET       RETRY      1000          10
ENOBUFS         RETRY      100           60
ENOTCONN        RETRY      100           10
EPIPE           RETRY      500           10
ESHUTDOWN       RETRY      1000          10
ETIMEDOUT       RETRY      1000          10
NODYNPORTS      RETRY      100           10
```

The `TCPERRS` file contains default responses to basic errors. To alter the instructions or add instructions for new errors, open the file in a text editor and change any of the values in the columns shown in Table 11-17.

Maintaining Updated Error Messages

The error, information, and warning messages that Oracle GoldenGate processes generate are stored in a data file named `ggmessage.dat` in the Oracle GoldenGate installation directory. The version of this file is checked upon process startup and must be identical to that of the process in order for the process to operate.

Resolving Oracle GoldenGate Errors

To get help with specific troubleshooting issues, go to My Oracle Support at `http://support.oracle.com` and search the Knowledge Base.

Trail File Management

The Extract process captures the changes from the transaction logs of the source system (database) into trail files that are consumed by other Oracle GoldenGate processes.

Extract can write into one or multiple sets of trail files. A trail is a sequence of files that are created and aged as needed. Processes that read a trail are:

- **Replicat**: Replicat reads the trail file received on the target deployment.
- **Distribution Service**: Extracts data from a local trail for further processing, if needed, and transfers it to the target system.
- **Receiver Service**: Receives the trail and transfers to Replicat, which reads the trail and applies change data to the target database.

You can create more than one trail to separate the data of different tables or applications, or to satisfy the requirements of a specific replication topology, such as a cascading topology. You link tables specified with a `TABLE` statement to a trail specified with an `EXTTRAIL` parameter statement in the Extract parameter file.

- Assign Storage for Oracle GoldenGate Trails
- Estimate Space for the Trail
- Add a Trail

Also see Using the LogDump Utility to Access Trail File Records.

Manage Trail Files

After data has been extracted, it must be processed into one or more trails, where it is stored for processing by another Oracle GoldenGate process. A trail is a sequence of files that are created and aged as needed. Processes that read a trail are:

- **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` or `RMTTRAIL` parameter statement in the Extract parameter file.

- Assign Storage for Oracle GoldenGate Trails
- Estimate Space for the Trail
- Add a Trail

See Using the LogDump Utility to Access Trail File Records.

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. Also see Purge Datastore.

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.

   ```
   [log volume in one hour] x [number of hours downtime] x .4 = trail disk
   space
   ```

   This equation uses a multiplier of 40 percent because only about 40 percent of the data in a transaction log is needed by Oracle GoldenGate.

> **Note:**
>
> This formula is a conservative estimate, and you should run tests once you have configured Oracle GoldenGate to determine exactly how much space you need.

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.

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.

To add a trail, issue the following command on the source system.

```
ADD {RMTTRAIL | EXTTRAIL} pathname, EXTRACT group [, MEGABYTES n]
```

Where:

- `RMTTRAIL` specifies a trail on a remote system.
- `EXTTRAIL` specifies a trail on the local system.
  - `EXTTRAIL` cannot be used for an Extract in PASSIVE mode.
  - `EXTTRAIL` must be used to specify a local trail.
- `pathname` 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\dirdat\rt. Oracle GoldenGate appends a serial number to each trail file as it is created during processing. Typically, trails are stored in the `dirdat` sub-directory of the Oracle GoldenGate directory.
- `EXTRACT` group specifies the name of the Extract group that writes to this trail. Only one Extract group can write to a trail.
- `MEGABYTES n` is an optional argument with which you can set the size, in megabytes, of each trail file (default is 100).

Example: Create a Local Trail

This example creates a local trail named `/ggs/dirdat/lt` for Extract group exte.

```
ADD EXTTRAIL /ggs/dirdat/lt, EXTRACT exte
```

Example: Create a Remote Trail

This example creates a trail named `c:\ggs\dirdat\rt` for Extract group finance, with each file sized at approximately 50 megabytes.

```
ADD RMTTRAIL c:\ggs\dirdat\rt, EXTRACT finance, MEGABYTES 200
```

Automate Maintenance Tasks

Use the **Tasks** tab on the Configuration page, to set up the following automated tasks.

Purging Trails

The Purge Trail page works the same way as the Manager `PURGEOLDEXTRACTS` parameter in the Classic Architecture. It allows you to purge trail files when Oracle GoldenGate has finished processing them. Automating this task ensures that the trail files are periodically deleted to avoid excessive consumption of disk space.

From the Tasks tab, when you select the Purge Trail page, it allows you to configure the Administration Service purge trail process.

1. Add a Purge Trail task by clicking the + sign .
2. Enter the **Operation Name** of the Administration Service task. The operation name is case sensitive. For example, you can create an operation with the name **TASK1** and another operation named **task1**.
3. Enter the trail path or trail name in the **Trail** field. The default trail file location is *$deployment_home*/lib/data.

   If non-default trail file locations are used for storing and retrieving trail files, then the full path along with the trail name must be added. An example of the full path of the trail file location would be: *$deployment-home*/lib/data/rep01/trail/et.
4. Click the **+** sign to add the trail to the **Selected Trails** list.
5. If you don't need to use checkpoints, disable the option **Use Checkpoints**. However, Oracle recommends using checkpoints. If you don't use checkpoints. the trail will be purged whether or not it has been consumed if the keep rule is met.
6. Set the **Keep Rule** value to specify the maximum number of hours, days, or number of files for which the Purge Trails task needs to be active.
7. Specify the number of hours or days when the purge trails task has to run, in the Purge Frequency field and click Submit.
8. Use the Purge Trails task table to edit or delete the task, as required.

   Also see PURGE EXTTRAIL.

Purging Tasks

You can automatically purge processes associated with an Administration Service.

From the Tasks tab, click Purge Tasks.

1. Enter the **Operation Name** that you need to set up for automatic purging.
2. Select the Extract or Replicat task (initial load process) **Process Name** for the operation. The list contains all processes so ensure that you select the correct task.
3. Select the Extract or Replicat task (initial load) **Process Type** for the operation.
4. If you enable **Use Stop Status**, the status of the task is used to perform the purge task.
5. Enter the hours or days after which you need to purge the process and click **Submit**.
6. Edit or delete the purge process task using the relevant icon from the Purge Tasks table.

Reporting Lag

You can manage lag reports from the Lag Report tab. To do so:

1. From the Tasks tab, click Lag Report.

2. The Action column contains all the options to delete, alter, refresh, and view the lag report task details.
3. Select the required option.
4. If you select the Alter Task option, you are presented with options to edit the lag report. The options are:
   - Enabled: To keep processing the lag report task.
   - Check Every (in minutes): To set a time interval to check the lag report.
   - Report: To log report for the task.
   - If Exceeds: To specify a threshold after which a warning would be initiated.
   - Warning: To allow a warning to be generated incase the lag threshold exceeds the specified limit.
   - When Exceeds: The lag threshold after which the warning is triggered.
5. Click Submit.

Admin Client Command Line Interface for Oracle GoldenGate Microservices

To start the Admin Client, you need to change the current working directory to the Oracle GoldenGate home directory (`OGG_HOME`).

For a complete list and description of commands available from the Admin Client, see About the Command Line Interfaces in the *Command Line Interface Reference for Oracle GoldenGate*.

About Admin Client

Admin Client is a command line utility. It uses the REST API published by the microservices to accomplish control and configuration tasks in an Oracle GoldenGate deployment.

Admin Client is a command line utility that can be used to created, modify, and remove Oracle GoldenGate processes and can be used in place of the MA web user interface.The Admin Client program is located in the `$OGG_HOME/bin` directory, where `$OGG_HOME` is the Oracle GoldenGate home directory.

If you need to automate the Admin Client connection with the deployment, you can use an Oracle Wallet to store the user credentials. The credentials stored must have the following characteristics:

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

> **Note:**
>
> To use the Admin Client for administration tasks, you need the user credentials that work with both the Service Manager and Administration Service.

To use the Admin Client, perform the following steps:

1. In Linux, set the `OGG_HOME`, and `PATH` environment variable to the following:

   ```
   export OGG_HOME=ogg_install_location
   ```

   ```
   export PATH=$OGG_HOME/bin:$PATH
   ```

   If you configure a secure deployment using SSL certificate files (`.pem` or `.der`), you must add the `OGG_CLIENT_TLS_CAPATH` environment variable. This is required to be able to connect to the deployment from Admin Client. This variable is used to specify the location where the certificate files are located on the host. For clients only needing to validate server certificates, the `OGG_CLIENT_TLS_CAPATH` environment variable should refer to a file containing a trusted CA Certificate that is shared with the server to which the client is expected to connect.

   ```
   export OGG_CLIENT_TLS_CAPATH = deployment_rootCA_certificate_location
   ```

   > **Note:**
   >
   > For Microsoft Windows, the default certificate file format is `.der` while all other platforms use `.pem` as the default format.

2. Run the command:

   ```
   [oracle]$ adminclient
   ```

   The output displays the Oracle GoldenGate Admin Client prompt, where you can connect to the deployment from the Admin Client:

   ```
   OGG (not connected) 1>
   ```

3. Connect to a deployment or to a proxy server from the Admin Client as a security user. This is the user you created while adding the deployment for your Oracle GoldenGate instance using OGGCA.

   ```
   CONNECT http(s)://localhost:port DEPLOYMENT deployment name AS security
   role user PASSWORD password
   ```

> **Note:**
>
> If your 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 - Connect to an Oracle GoldenGate Service Manager
        |CONNECT server-url [ DEPLOYMENT deployment-name]
        |[ ( AS deployment-credentials-name|
        | USER  deployment-user-name )
        |[PASSWORD deployment-password] ]
        |[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 Oracle GoldenGate deployment user to connect to the deployment from the Admin Client:

```
ADD CREDENTIALS admin USER ggadmin PASSWORD oggadmin-A1
```

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 oggadmin-A2
```

Output:

```
2019-02-14T00:35:48Z  INFO OGG-15114  Credential store altered.


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>
```

If the credentials are invalid for a proxy connection, then an error similar to the following error occurs:

```
ERROR: Proxy server user name 'proxyadmin' or password is incorrect.
```

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 11-37 OBEY command file

```
ADD EXTRACT myext, TRANLOG, BEGIN now
START EXTRACT myext

ADD REPLICAT myrep, EXTTRAIL /ggs/dirdat/aa
START REPLICAT myrep

INFO EXTRACT myext, DETAIL
INFO REPLICAT myrep, DETAIL
```

The following example illustrates an `OBEY` command file for use with the `OBEY` command. It creates and starts Extract and Replicat groups and retrieves processing information.

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* is: KILL, START, or STOP
- *wildcard specification* is a wildcard specification for the names of the process groups that you want to affect with the command. The command affects every Extract and Replicat group that satisfies the wildcard. Oracle GoldenGate supports up to 100,000 wildcard entries.

Deleting Extract and Replicat

This section contains basic directions for deleting Extract and Replicat processes.

To Delete an Extract Group

1. Connect to the deployment from the Admin Client.
2. Issue the DBLOGIN command as the Extract database user (or a user with the same privileges). You can use either of the following commands, depending on whether a local credential store exists.

   ```
   DBLOGIN [SOURCEDB dsn] {USERID user, PASSWORD password
   [encryption_options] | USERIDALIAS alias [DOMAIN domain]}
   ```

3. Stop the Extract process.

   ```
   STOP EXTRACT group_name
   ```

4. Issue the following command.

   ```
   DELETE EXTRACT group_name
   ```

5. (Oracle) Unregister the Extract group from the database.

   ```
   UNREGISTER EXTRACT group_name,database_name
   ```

To Delete a Replicat Group

1. Stop the Replicat process.

   ```
   STOP REPLICAT group_name
   ```

2. Issue one of the following commands to log into the database.

```
DBLOGIN [SOURCEDB dsn] {USERID user, PASSWORD password
[encryption_options] | USERIDALIAS alias [DOMAIN domain]}
```

Where:

- `SOURCEDB dsn` supplies the data source name, if required as part of the connection information.
- `USERID user`, `PASSWORD password` specifies an explicit database login credential.
- `USERIDALIAS alias [DOMAIN domain]` specifies an alias and optional domain of a credential that is stored in a local credential store.
- *encryption_options* is one of the options that encrypt the password.

3. Issue the following command to delete the group.

```
DELETE REPLICAT group_name
```

Deleting a Replicat group preserves the checkpoints in the checkpoint table (if being used). Deleting a process group also preserves the parameter file. You can create the same group again, using the same parameter file, or you can delete the parameter file to remove the group's configuration permanently.

Specifying Object Names in Oracle GoldenGate Input

The following rules apply when specifying object names in parameter files (such as in `TABLE` and `MAP` statements), column-conversion functions, commands, and in other input.

Specifying Filesystem Path Names in Parameter Files on Windows Systems

On Windows systems, if the name of any directory in a filesystem path name begins with a number, the path must be specified with forward slashes, not backward slashes, when listing that path in Oracle GoldenGate input, such as parameter files or commands. This requirement prevents Oracle GoldenGate from interpreting the name as an octal escape sequence. For example, the following paths contain a directory named `\2014` that will be interpreted as the octal sequence `\201`:

```
C:\ogg\2014\install\dirdat\aa
C:\ogg\install\2014\dirdat\aa
```

The preceding path can be used with forward slashes as follows:

```
C:/ogg/2014/install/dirdat/aa
C:/ogg/install/2014/dirdat/aa
```

For more information, see Support for Escape Sequences.

Supported Database Object Names

Object names in parameter files, command, and other input can be any length and in any supported character set. For supported character sets, see Supported Character Sets.

Oracle GoldenGate supports most characters in object and column names. Specify object names in double quote marks if they contain special characters such as white spaces or symbols.

The following lists of supported and non-supported characters covers all databases supported by Oracle GoldenGate; a given database platform may or may not support all listed characters.

Supported Special Characters

Oracle GoldenGate supports all characters that are supported by the database, including the following special characters. Object names that contain these special characters must be enclosed within double quotes in parameter files.

Character	Description
/	Forward slash (See Specifying Names that Contain Slashes)
*	Asterisk (Must be escaped by a backward slash when used in parameter file, as in: *)
?	Question mark (Must be escaped by a backward slash when used in parameter file, as in: \?)
@	At symbol (Supported, but is often used as a resource locator by databases. May cause problems in object names)
#	Pound symbol
$	Dollar symbol
%	Percent symbol (Must be %% when used in parameter file)
^	Caret symbol
( )	Open and close parentheses
_	Underscore
-	Dash
<space>	Space

Non-supported Special Characters

The following characters are not supported in object names and non-key column names.

Character	Description
\	Backward slash (Must be \\ when used in parameter file)
{ }	Begin and end curly brackets (braces)
[ ]	Begin and end brackets
=	Equal symbol
+	Plus sign
!	Exclamation point
~	Tilde
|	Pipe
&	Ampersand
:	Colon
;	Semi-colon

Character	Description
,	Comma
''	Single quotes
""	Double quotes
'	Accent mark (Diacritical mark)
.	Period
<	Less-than symbol (or beginning angle bracket)
>	Greater-than symbol (or ending angle bracket)

Specifying Names that Contain Slashes

If a table name contains a forward-slash character (/) in any part of its name, that name component must be enclosed within double quotes unless the object name is from an IBM i platform . The following are some examples:

```
"c/d"
"/a".b
a."b/"
```

If the name contains a forward slash that is not enclosed within double quotes, Oracle GoldenGate treats it as a name that originated on the IBM i platform (from a DB2 for i database). The forward slash in the name is interpreted as a separator character.

Qualifying Database Object Names

Object names must be fully qualified in the parameter file. This means that every name specification must be qualified, not only those supplied as input to Oracle GoldenGate parameter syntax, but also names in a SQL procedure or query that is supplied as `SQLEXEC` input, names in user exit input, and all other input supplied in the parameter file.

Oracle GoldenGate supports two-part and three-part object names, as appropriate for the database.

Two-part Names

Most databases require only two-part names to be specified, in the following format:

`owner.object`

For example: `HR.EMP`

Where:

`owner` is a schema or database, depending on how the database defines a logical namespace that contains database objects. *`object`* is a table or other supported database object.

The databases for which Oracle GoldenGate supports two-part names are as follows, shown with their appropriate two-part naming convention:

- Db2 for i: *`schema.object`* and *`library/file(member)`*
- Db2 LUW: *`schema.object`*
- Db2 on z/OS: *`schema.object`*

- MySQL: *database.object*
- Oracle Database (non-CDB databases): *schema.object*
- SQL Server: *schema.object*
- Teradata: *database.object*

Three-part Names

Oracle GoldenGate supports three-part names for the following databases:

- Oracle container databases (CDB)

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*;
```

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 11-18 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 11-18 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.

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 `TABn`, 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 DEFGEN. 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_TEST`*n*. 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_TEST`*n* 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.

Differentiating Case-Sensitive Column Names from Literals

By default, Oracle GoldenGate follows SQL-92 rules for specifying column names and literals. In Oracle GoldenGate parameter files, conversion functions, user exits, and commands, case-sensitive column names must be enclosed within double quotes if the database requires quotes around a name to support case-sensitivity. For example:

```
"columnA"
```

Case-sensitive column names in databases that do not require quotes to enforce case-sensitivity must be specified as they are stored in the database. For example:

```
ColumnA
```

Literals must be enclosed within single quotes. In the following example, `Product_Code` is a case-sensitive column name in an Oracle database, and the other strings are literals.

```
@CASE ("Product_Code", 'CAR', 'A car', 'TRUCK', 'A truck')
```

Creating a Parameter File Using Admin Client

To create a parameter file, run the `EDIT PARAMS` command from the Admin Client. When you create a parameter file with `EDIT PARAMS`, it is saved to the `dirprm` sub-directory of the Oracle GoldenGate directory.
You can create a parameter file in a directory other than dirprm, but you also must specify the full path name with the `PARAMS` option of the `ADD EXTRACT` or `ADD REPLICAT` command when you create your process groups. After pairing with an Extract or Replicat group, a parameter file

must remain in its original location for Oracle GoldenGate to operate properly after processing has started.
The `EDIT PARAMS` command launches the following text editors in Admin Client:

- Notepad on Microsoft Windows systems.
- The vi editor on UNIX and Linux systems. Db2 for i only supports vi when connected with SSH or xterm. For more information, see Creating a Parameter File with a Text Editor.

> **Note:**
>
> You can change the default editor through Admin Client by using the `SET EDITOR` command.

1. Run the Admin Client.
2. Connect to the Admin Client using the `CONNECT` command.
3. In Admin Client, issue the following command to open the default text editor:

   ```
   EDIT PARAMS group_name
   ```

   In this code snippet:
   group_name is the name of the Extract or Replicat group for which the file is being created. The name of an Extract or Replicat parameter file must match that of the process group.

   The following creates or edits the parameter file for an Extract group named `exte`:

   ```
   EDIT PARAMS exte
   ```

4. Using the editing functions of the text editor, enter as many comment lines as you want to describe this file, making certain that each comment line is preceded with two hyphens (--).
5. On non-commented lines, enter the Oracle GoldenGate parameters, starting a new line for each parameter statement.

   Oracle GoldenGate parameters have the following syntax:

   ```
   PARAMETER_NAME argument [,option] [&]
   ```

   Where:

   - `PARAMETER_NAME` is the name of the parameter.
   - `argument` is a required argument for the parameter. Some parameters take arguments, but others do not. Commas between arguments are optional.

     ```
     EXTRACT exte
      USERIDALIAS ggadmin
      ENCRYPT AES192 KEYNAME mykey ENCRYPTTRAIL AES 192
      EXTTRAIL /north/ea, PURGE CUSEREXIT userexit.dll MyUserExit,
     INCLUDEUPDATEBEFORES, & PARAMS "init.properties"
      TABLE hr.employees;
     ```

   - `[,option]` is an optional argument.

- `[&]` is required at the end of each line in a multi-line parameter statement, as in the `CUSEREXIT` parameter statement in the previous example. The exceptions are the following, which can accept, but do not require the ampersand because they terminate with a semicolon:
  - `MAP`
  - `TABLE`
  - `SEQUENCE`
  - `FILE`
  - `QUERY`

6. Save and close the file.

Creating a Parameter File with a Text Editor

You can create a parameter file outside Admin Client by using a text editor, but make certain to:

- Save the parameter file with the name of the Extract or Replicat group that owns it. Use the `.prm` file extension. For example: `exte.prm`.
- Save the parameter file in the `dirprm` directory of the Oracle GoldenGate home directory.

Validating a Parameter File

You can validate the parameter file from the Administration Service web interface. You can validate the Extract and Replicat parameters from the **Reports** tab. To access the **Reports** tab:

1. From Extract or Replicat section of the Administration Service Overview Page, click **Action** and then click **Details**.
2. Click the **Reports** tab to view the report for Extract and Replicat parameters, error log, and other information.

See Access Extract Details to learn how to check and edit the Extract parameters. See Access Replicat Details to learn about editing Replicat parameter files. Also see Additional Parameter Options for Integrated Replicat

You can also use the `checkprm` validation native command is run from the command line and give an assessment of the specified parameter file, with a configurable application and running environment. It can provide either a simple PASS/FAIL or with additional details about how the values of each parameter are stored and interpreted.

The `CHECKPRM` executable file can be found in the `$OGG_HOME/bin` directory of Microservices Architecture. See checkprm in the *Parameters and Functions Reference for Oracle GoldenGate*. The input to `checkprm` is case insensitive. If a value string contains spaces, it does not need to be quoted because checkprm can recognize meaningful values. If no mode is specified to `checkprm`, then all parameters applicable to any mode of the component will be accepted.
The output of `checkprm` is assembled with four possible sections:

- help messages
- pre-validation error
- validation result
- parameter details

A pre-validation error is typically an error that prevents a normal parameter validation from executing, such as missing options or an inaccessible parameter file. If an option value is specified incorrectly, a list of possible inputs for that option is provided. If the result is `FAIL`, each error is in the final result message. If the result is `PASS`, a message that some of the parameters are subject to further runtime validation. The parameter detailed output contains the validation context, and the specified parameters. The parameter and options are printed with proper indentation to illustrate these relationships.

See CHECKPARAMS parameter.

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 scott?TABNAME, TARGET tiger ACCOUNT_TARG;
```

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 11-38 Parameter substitution on Windows

```
C:\GGS> set EXTFILE=C:\ggs\extfile
C:\GGS> set TABNAME=PROD.ACCOUNTS
C:\GGS> replicat paramfile c:\ggs\dirprm\parmfl
```

Example 11-39 Parameter substitution on UNIX (Korn shell)

```
$ EXTFILE=/ggs/extfile
$ export EXTFILE
$ TABNAME=PROD.ACCOUNTS
$ export TABNAME
$ replicat paramfile ggs/dirprm/parmfl
```

UNIX is case-sensitive, so the parameter declaration in the parameter file must be the same case as the shell variable assignments.

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 11-19 Macro Definition Arguments

Argument	Description
`MACRO`	Required. Indicates the start of an Oracle GoldenGate macro definition.
`#macro_name`	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 `#macro1` and `#param1`. A macro or parameter name can be one word consisting of letters and numbers, or both. Special characters, such as the underscore character (_) or hyphen (-), can be used. Some examples of macro names are: `#mymacro`, `#macro1`, `#macro_1`, `#macro-1`, `#macro$`. Some examples of parameter names are `#sourcecol`, `#s`, `#col1`, and `#col_1`. To avoid parsing errors, the macro character cannot be used as the first character of a macro name. For example, `##macro` is invalid. If needed, you can change the macro character by using the `MACROCHAR` parameter. See Reference for Oracle GoldenGate for Windows and UNIX. Macro and parameter names are not case-sensitive. Macro or parameter names within quotation marks are ignored.
`PARAMS (#p1, #p2)`	Optional definition of input parameters. Specify a comma-separated list of parameter names and enclose it within parentheses. Each parameter must be referenced in the macro body where you want input values to be substituted. You can list each parameter on a separate line to improve readability (making certain to use the open and close parentheses to enclose the parameter list). See Call a Macro that Contains Parameters for more information.
`BEGIN`	Begins the macro body. Must be specified before the macro body.

Table 11-19 (Cont.) Macro Definition Arguments

Argument	Description

macro_body	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:
	`COL1 = COL2`
	• Complex parameter statements with parameter substitution as in:
	`MAP #o.#t, TARGET #o.#t, KEYCOLS (#k), COLMAP (USEDEFAULTS);`
	• Invocations of other macros, as in:
	`#colmap (COL1, #sourcecol)`

`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 11-20 Syntax Elements for Calling a Macro

Argument	Description

target =	Optional. Specifies the target to which the results of the macro are assigned or mapped. For example, `target` can be used to specify a target column in a `COLMAP` statement. In the following call to the `#make_date` macro, the column `DATECOL1` is the target and will be mapped to the macro results.
	
	`DATECOL1 = #make_date (YR1, MO1, DAY1)`
	
	Without a target, the syntax to call `#make_date` is:
	
	`#make_date (YR1, MO1, DAY1)`

macro_name	The name of the macro that is being called, for example: `#make_date`.
`( val[, ...])`	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 ().
`( val | {val, val, ...} )[, ...]`	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, {DEPTNO1, DEPTNO2, DEPTNO3})`. The ability to substitute a block of values for any given parameter add flexibility to the macro definition and its usability in the Oracle GoldenGate configuration.
- Calls to other macros, such as: `#macro (#mycalc (col2, 100), #total)`. In this example, the `#mycalc` macro is called with the input values of `col2` and `100`.

Oracle GoldenGate substitutes parameter values within the macro body according to the following rules.

1. The macro processor reads through the macro body looking for instances of parameter names specified in the `PARAMS` statement.
2. For each occurrence of the parameter name, the corresponding parameter value specified during the call is substituted.
3. If a parameter name does not appear in the `PARAMS` statement, the macro processor evaluates whether or not the item is, instead, a call to another macro. (See Calling Other Macros from a Macro.) If the call succeeds, the nested macro is executed. If it fails, the whole macro fails.

Example 11-40 Using Parameters to Populate a MAP Statement

The following macro definition specifies three parameter that must be resolved. The parameters substitute for the names of the table owner (parameter `#o`), the table (parameter `#t`), and the `KEYCOLS` columns (parameter `#k`) in a `MAP` statement.

```
MACRO #macro1  PARAMS ( #o, #t, #k )  BEGIN  MAP #o.#t, TARGET #o.#t, KEYCOLS
(#k), COLMAP (USEDEFAULTS); END;
```

Assuming a table in the `MAP` statement requires only one `KEYCOLS` column, the following syntax can be used to call `#macro1`. In this syntax, the `#k` parameter can be resolved with only one value.

```
#macro1 (SCOTT, DEPT, DEPTNO1)
```

To call the macro for a table that requires two `KEYCOLS` columns, the curly brackets are used as follows to enclose both of the required values for the column names:

```
#macro1 (SCOTT, DEPT, {DEPTNO1, DEPTNO2})
```

The `DEPTNO1` and `DEPTNO2` values are passed as one argument to resolve the `#t` parameter. Tables with three or more `KEYCOLS` can also be handled in this manner, using additional values inside the curly brackets.

Example 11-41 Using a Macro to Perform Conversion

In this example, a macro defines the parameters `#year`, `#month`, and `#day` to convert a proprietary date format.

```
MACRO #make_date
PARAMS (#year, #month, #day)
BEGIN
@DATE ('YYYY-MM-DD', 'CC', @IF (#year < 50, 20, 19), 'YY', #year, 'MM',
```

```
#month, 'DD', #day)
END;
```

The macro is called in the `COLMAP` clause:

```
MAP sales.acct_tab, TARGET sales.account,
COLMAP
(
targcol1 = sourcecol1,
datecol1 = #make_date(YR1, MO1, DAY1),
datecol2 = #make_date(YR2, MO2, 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',
MO1, 'DD', DAY1),
datecol2 = @DATE ('YYYY-MM-DD', 'CC', @IF (YR2 < 50, 20, 19),'YY', YR2, 'MM',
MO2, 'DD', DAY2)
);
```

Call a Macro without Input Parameters

To call a macro without input parameters, the call statement must supply the open and close parentheses, but without any input values: `#macro ()`.

The following macro is defined without input parameters. The body contains frequently used parameters.

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

This macro is called as follows:

```
#option_defaults ()
IGNOREUPDATES
MAP owner.srctab, TARGET owner.targtab;

#option_defaults ()
MAP owner.srctab2, TARGET owner.targtab2;
```

The macro expands as follows:

```
GETINSERTS
GETUPDATES
GETDELETES
INSERTDELETES
IGNOREUPDATES
MAP owner.srctab, TARGET owner.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 `ERCALLBACK` function that is to be called from the user exit routine. The user exit must explicitly load the callback function at run-time using the appropriate Windows API calls.

4. Include the CUSEREXIT parameter in your Extract or Replicat parameter file. This parameter accepts the name of the shared object or DLL and the name of the exported routine that is to be called from Extract or Replicat. You can specify the full path of the shared object or DLL or let the operating system's standard search strategy locate the shared object.

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

Where:

- *DLL* is a Windows DLL and *shared_object* is a UNIX shared object that contains the user exit function.
- INCLUDEUPDATEBEFORES gets before images for UPDATE operations.
- PARAMS *'startup_string'* supplies a startup string, such as a startup parameter.

Example 11-42 Example of Base Syntax, UNIX

```
CUSEREXIT eruserexit.so MyUserExit
```

Example 11-43 Example Base Syntax, Windows

```
CUSEREXIT eruserexit.dll MyUserExit
```

Supporting Character-set Conversion in User Exits

To maintain data integrity, a user exit needs to understand the character set of the character-type data that it exchanges with an Oracle GoldenGate process. Oracle GoldenGate user exit logic provides globalization support for:

- character-based database metadata, such as the names of catalogs, schemas, tables, and columns
- the values of character-type columns, such as CHAR, VARCHAR2, CLOB, NCHAR, NVARCHAR2, and NCLOB, as well as string-based numbers, date-time, and intervals.

Properly converting between character sets allows column data to be compared, manipulated, converted, and mapped properly from one type of database and character set to another. Most of this processing is performed when the EXIT_CALL_PROCESS_RECORD call type is called and the record buffer and other record information is made available through callback routines.

The user exit has its own session character set. This is defined by the GET_SESSION_CHARSET and SET_SESSION_CHARSET callback functions. The caller process provides conversion between character sets if the character set of the user exit is different from the hosting context of the process.

To enable this support in user exits, there is the GET_DATABASE_METADATA callback function code. This function enables the user exit to get database metadata, such as the locale and the character set of the character-type data that it exchanges with the process that calls it (Extract, data pump, Replicat). It also returns how the database treats the case-sensitivity of object names, how it treats quoted and unquoted names, and how it stores object names.

For more information about these components, see Reference for Oracle GoldenGate for Windows and UNIX.

Using Macros to Check Name Metadata

The object name that is passed by the user exit API is the exact name that is encoded in the user-exit session character set, and exactly the same name that is retrieved from the database. If the user exit compares the object name with a literal string, the user exit must retrieve the database locale and then normalize the string so that it is compared with the object name in the same encoding.

Oracle GoldenGate provides the following macros that can be called by the user exit to check the metadata of database object names. For example, a macro can be used to check whether a quoted table name is case-sensitive and whether it is stored as mixed-case in the database server. These macros are defined in the `usrdecs.h` file.

Table 11-21 Macros for metadata checking

Macro	What it verifies
`supportsMixedCaseIdentifiers( nameMeta, DbObjType )`	Whether the database treats a mixed-case unquoted name of a specified data type as case-sensitive and stores the name in mixed case.
`supportsMixedCaseQuotedIdentifiers( nameMeta, DBObjType )`	Whether the database treats the mixed-case quoted name of a specified data type as case-sensitive and stores the name in mixed case.
`storesLowerCaseIdentifiers( nameMeta, DbObjType )`	Whether the database treats the mixed-case unquoted name of a specified data type as case-insensitive and stores the name in lower case.
`storesLowerCaseQuotedIdentifiers( nameMeta, DbObjType )`	Whether the database treats the mixed-case quoted name of a specified data type as case-insensitive and stores the name in lower case.
`storesMixedCaseIdentifiers( nameMeta, DbObjType )`	Whether the database treats the mixed-case unquoted name of a specified data type as case-insensitive and stores the name in mixed case.
`storesMixedCaseQuotedIdentifiers( nameMeta, DbObjType )`	Whether the database treats the mixed-case quoted name of a specified data type as case-insensitive and stores the name in mixed case.
`storesUpperCaseIdentifiers( nameMeta, DbObjType )`	Whether the database treats the mixed-case unquoted name of a specified data type as case-insensitive and stores the name in upper case.
`storesUpperCaseQuotedIdentifiers( nameMeta, DbObjType )`	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 11-22 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. Tailing spaces are trimmed.	"abc" encoded in user exit session character set. NOT NULL terminated. Tailing 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. Tailing spaces are trimmed by default unless the GLOBALS parameter NOTRIMSPACES is specified.
VARCHAR2 "abc"	2-byte null indicator + 2-byte length info + column value	"abc" encoded in ASCII or EBCDIC. NULL terminated. No trimming.	"abc" encoded in user exit session character set. NOT NULL terminated. No trimming.
NVARCHAR2 0061 0062 0063 0020	2-byte null indicator + 2-byte length info + column value	"abc" (encoded in UTF8) or truncated at the first byte, depending on whether NVARCHAR2 is treated as UTF-8. NULL terminated. No trimming.	"abc"encoded in user exit session character set. NOT NULL terminated. No trimming.
CLOB	2-byte null indicator + 2-byte length info + column value	Similar to VARCHAR2, but only output up to 4K bytes. NULL Terminated. No trimming.	Similar to VARCHAR2, but only output data requested in user exit session character set. NOT NULL terminated. No trimming.
NCLOB	2-byte null indicator + 2-byte length info + column value	Similar to NVARCHAR2, but only output up to 4K bytes. NULL terminated. No trimming.	Similar to NVARCHAR2, but only output data requested in user exit session character set. NOT NULL terminated. No trimming.
NUMBER 123.89	2-byte null indicator + 2-byte length info + column value	"123.89" encoded in ASCII or EBCDIC. NULL terminated.	"123.89" encoded in user exit session character set. NOT NULL terminated.
DATE 31-May-11	2-byte null indicator + 2-byte length info + column value	"2011-05-31" encoded in ASCII or EBCDIC. NULL terminated.	"2011-05-31" encoded in user exit session character set. NOT NULL terminated.
TIMESTAMP 31-May-11 12.00.00 AM	2-byte null indicator + 2-byte length info + column value	"2011-05-31 12.00.00 AM" encoded in ASCII or EBCDIC. NULL terminated.	"2011-05-31 12.00.00 AM" encoded in user exit session character set. NOT NULL terminated.

Table 11-22 (Cont.) column_value_mode_matrix Meanings

Data Type	EXIT_FN_RAW_FORMAT	EXIT_FN_CHAR_FORMAT	EXIT_FN_CNVTED_SESS_FORMAT
`Interval Year to Month or Interval Day to Second`	2-byte null indicator + 2-byte length info + column value	NA	NA
`RAW`	2-byte null indicator + 2-byte length info + column value	2-byte null indicator + 2-byte length info + column value	2-byte null indicator + 2-byte length info + column value

Upgrading User Exits

The `usrdecs.h` file is versioned to allow backward compatibility with existing user exits when enhancements or upgrades, such as new functions or structural changes, are added to a new Oracle GoldenGate release. The version of the `usrdecs.h` file is printed in the report file at the startup of Replicat or Extract.

To use new user exit functionality, you must recompile your routines to include the new `usrdecs` file. Routines that do not use new features do not need to be recompiled.

Viewing Examples of How to Use the User Exit Functions

Oracle GoldenGate installs the following sample user exit files into the `UserExitExamples` directory of the Oracle GoldenGate installation directory:

- `exitdemo.c` shows how to initialize the user exit, issue callbacks at given exit points, and modify data. It also demonstrates how to retrieve the fully qualified table name or a specific metadata part, such as the name of the catalog or container, or the schema, or just the unqualified table name. In addition, this demo shows how to process DDL data. The demo is not specific to any database type.
- `exitdemo_utf16.c` shows how to use UTF16-encoded data (both metadata and column data) in the callback structures for information exchanged between the user exit and the caller process.
- `exitdemo_more_recs.c` shows an example of how to use the same input record multiple times to generate several target records.
- `exitdemo_lob.c` shows an example of how to get read access to LOB data.
- `exitdemo_pk_befores.c` shows how to access the before and after image portions of a primary key update record, as well as the before images of regular updates (non-key updates). It also shows how to get target row values with `SQLEXEC` in the Replicat parameter file as a means for conflict detection. The resulting fetched values from the target are mapped as the target record when it enters the user exit.

Each directory contains the `*.c` files as well as makefiles and a `readme.txt` file.

12 Performance

Learn about different techniques available with Oracle GoldenGate to carry out performance monitoring and tuning activities.

Monitor

Learn about monitoring Oracle GoldenGate processes for performance and error handling.

Commands Used for Monitoring

You can view information about Extract and Replicat groups from the Oracle GoldenGate MA web interface at various levels. Another alternative is to use the command line interface to monitor various processes.

See Monitor Processes from the Performance Metrics Service.

To learn about command syntax, usage, and examples, see the *Command Line Interface Reference* for Oracle GoldenGate.

Command	What it Shows
`INFO {EXTRACT | REPLICAT} group [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} group`	Displays statistics on processing volume, such as number of operations performed.
`STATUS {EXTRACT | REPLICAT} group`	Displays the run status (starting, running, stopped, abended) for Extract and Replicat processes.
`LAG {EXTRACT | REPLICAT} group`	Displays the latency between last record processed and timestamp in the data source.
`INFO {EXTTRAIL | RMTTRAIL } trail`	Displays the name of associated process, position of last data processed, maximum file size.
`SEND {EXTRACT | REPLICAT } group`	Depending on the process and selected options, returns information about memory pool, lag, TCP statistics, long-running transactions, process status, recovery progress, and more.

Command	What it Shows
`VIEW REPORT group`	Shows contents of the discard file or process report.
`VIEW GGSEVT`	Shows contents of the Oracle GoldenGate error log.

COMMAND ER *wildcard*	Information dependent on the `COMMAND` type:
	`INFO`
	`LAG`
	`SEND`
	`STATS`
	`STATUS`
	`wildcard`
	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.
`INFO EXTTRAIL`	Retrieves configuration information for a local trail. It shows the name of the trail, the Extract that writes to it, the position of the last data processed, and the assigned maximum file size.

Command	What it Shows
`INFO RMTTRAIL`	Retrieves configuration information for a remote trail. It shows the name of the trail, the Extract that writes to it, the position of the last data processed, and the assigned maximum file size.
`INFO ER`	Retrieves information on multiple Extract and Replicat groups as a unit.
`INFO CHECKPOINTTABLE`	Confirms the existence of a checkpoint table and view the date and time that it was created.
`INFO CREDENTIALS`	Retrieves a list of credentials.
`INFO ENCRYPTIONPROFILE`	Returns information about the encryption profiles available with the Service Manager.
`INFO HEARTBEATTABLE`	Displays information about the heartbeat tables configured in the database.
`INFO AUTHORIZATIONPROFILE`	Lists all the authorization profiles in a deployment or information on a specific authorization profile for a specific deployment.
`INFO MASTERKEY`	Displays the contents of a currently open master-key wallet. If a wallet store does not exist, a new wallet store file is created. This wallet store file is then used to host different encrypted keys as they are created.
`INFO PROFILE`	Returns information about managed process profiles.
`INFO RECVPATH`	Returns information about a target-initiated distribution path in the Receiver Service. Before you run this command, ensure that the Receiver Service is running.
`INFO SCHEMATRANDATA`	Valid for Oracle database only. Determine whether Oracle schema-level supplemental logging is enabled for the specified schema or if any instantiation information is available. Use the `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.

Command	What it Shows
`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 Processes from the Performance Metrics Service

The Performance Metrics Service uses the metrics service to collect and store instance deployment performance results. When you arrive at the Performance Metrics Service Overview page, you see all the Oracle GoldenGate processes in their current state.

You can click a process to view its performance metrics. You can also access service messages and status change details from this page.

Here's a general overview of the tasks that you can perform from this page.

Task	Description
Review Messages	Review Messages from Messages Overview Tab from the Messages Overview tab.
Review Status Changes	Click the Review Status Changes tab to review changes in status of a service.

Topics:

Review Messages from Messages Overview Tab

Messages from the Services are displayed in Performance Metrics Service Overview page.

To review the messages sent or received, do the following:

1. From the Service Manager, click **Performance Metrics Service**.

   The Performance Metrics Service Overview page is displayed.

2. Click the **Messages Overview** tab (if it's not already selected) to see a drill down into all the service messages.

   Scroll through the list of messages or search for a specific message by entering the text in the message.

3. Click **Refresh** to get a synchronized real-time list of messages before you start searching. You can also change the page size to view more or fewer messages.

Review Status Changes

Real-time status changes to microservices can be monitored from the Performance Metrics Service Status Changes Overview tab.

Status change messages show the date, process name, and its status, which could be running, starting, stopped, or killed.

To view status changes, click **Performance Metrics Service** from the Service Manager home page, and then click the **Status Changes Overview** tab. A list of status change messages from the service appears.

If you are searching for specific messages, you can use the search but make sure you click **Refresh** before you search to ensure that you get the updated status for services.

Note that the search messages appear in different colors to differentiate critical and informational messages.

Purge Datastore

You can change the datastore retention and purge it from the Performance Metrics Service **Monitoring Commands** tab.

To view status changes, click Performance Metrics Service from the Service Manager home page, and then click the **Monitoring Commands** tab.

The current process retention (in days) is displayed.

You can enter the number of retention days or use the sliding icon to set the new period from 1 to 365 days, then **Execute** to activate the purge. The details of the purge are also displayed.

Protocols for Performance Monitoring for Different Operating Systems

Oracle GoldenGate uses Unix Domain Sockets (UDS) for UNIX-based systems to send monitoring points from Extract, Replicat, and other processes to the Performance Monitoring Service of the deployment.

UDS is available with Oracle and non-Oracle databases. The UDS file is located in the `$OGG_HOME/var/temp` directory of the deployment.

For Oracle GoldenGate 21c, UDS is the only communication protocol for Performance Metrics Service. For Windows and other operating systems that don't support UDS, UDP is the default parameter.

Monitoring 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.

Topics:

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 ER 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.

Topics:

Monitoring an Extract Recovery

If Extract abends when a long-running transaction is open, it can seem to take a long time to recover when it is started again. To recover its processing state, Extract must search back through the online and archived logs (if necessary) to find the first log record for that long-running transaction. The farther back in time that the transaction started, the longer the recovery takes, in general, and Extract can appear to be stalled.

To confirm that Extract is recovering properly, use the `SEND EXTRACT` command with the `STATUS` option. One of the following status notations appears, and you can follow the progress as Extract changes its log read position over the course of the recovery.

In recovery[1]
Extract is recovering to its checkpoint in the transaction log. This implies that it is reading from either the BR checkpoint files and then archived/online logs, or reading from Recovery Checkpoint in archived/online log.

In recovery[2]
Extract is recovering from its checkpoint to the end of the trail. This implies that a recovery marker is appended to the output trail when the last transaction was not completely written then rewriting the transaction.

Recovery complete
The recovery is finished, and normal processing will resume.

Heartbeat Table End-To-End Replication Flow

The end-to-end replication process for heartbeat tables relies on using the Oracle GoldenGate trail format. The process is as follows:

Add a heartbeat table to each of your databases with the `ADD HEARTBEATTABLE` command. Add the heartbeat table to all source and target instances and then restart existing Oracle GoldenGate processes to enable heartbeat functionality. Depending on the database, you may or may not be required to create or enable a job to populate the heartbeat table data. See the following sample:

```
DBLOGIN USERIDALIAS alias  [DOMAIN domain]|[SYSDBA | SQLID sqlid]
[SESSIONCHARSET character_set]}


ADD HEARTBEATTABLE
```

(Optional) For Oracle Databases, you must ensure that the Oracle `DBMS_SCHEDULER` is operating correctly as the heartbeat update relies on it. You can query the `DBMS_SCHEDULER` by issuing:

```
SELECT START_DATE, LAST_START_DATE, NEXT_RUN_DATE
FROM DBA_SCHEDULER_JOBS
```

Where `job_name ='GG_UPDATE_HEARTBEATS';`
Then look for valid entries for `NEXT_RUN_DATE`, which is the next time the scheduler will run. If this is a timestamp in the past, then no job will run and you must correct it.
A common reason for the scheduler not working is when the parameter `job_queue_processes` is set too low (typically zero). Increase the number of `job_queue_processes` configured in the database with the `ALTER SYSTEM SET JOB_QUEUE_PROCESSES = ##;` command where `##` is the number of job queue processes.

Run an Extract, which on receiving the logical change records (LCR) checks the value in the `OUTGOING_EXTRACT` column.

- If the Extract name matches this value, the `OUTGOING_EXTRACT_TS` column is updated and the record is entered in the trail.
- If the Extract name does not match then the LCR is discarded.

- If the OUTGOING_EXTRACT value is NULL, it is populated along with OUTGOING_EXTRACT_TS and the record is entered in the trail.

The Distribution Service on reading the record, checks the value in the OUTGOING_ROUTING_PATH column. This column has a list of distribution paths.
If the value is NULL, then the column is updated with the current group name (and path if this is a Distribution Service),"*", update the OUTGOING_ROUTING_TS column, and the record is written into its target trail file.
If the value has a "*" in the list, then replace it with *group name[:pathname],"*"'*, update the OUTGOING_ROUTING_TS column, and the record is written into its target trail file. When the value does not have a 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:

```
REMOTE_DATABASE         = LOCAL_DATABASE
INCOMING_EXTRACT         = OUTGOING_EXTRACT
INCOMING_ROUTING_PATH      = OUTGOING_ROUTING_PATH with "*"    removed
INCOMING_REPLICAT        = @GETENV ("GGENVIRONMENT", "GROUPNAME")
INCOMING_HEARTBEAT_TS    = HEARTBEAT_TIMESTAMP
INCOMING_EXTRACT_TS    = OUTGOING_EXTRACT_TS
INCOMING_ROUTING_TS    = OUTGOING_ROUTING_TS
INCOMING_REPLICAT_TS          = @DATE ('UYYYY-MM-DD
HH:MI:SS.FFFFFF','JTSLCT',@GETENV ('JULIANTIMESTAMP'))
LOCAL_DATABASE    = REMOTE_DATABASE
OUTGOING_EXTRACT    = INCOMING_EXTRACT
OUTGOING_ROUTING_PATH    = INCOMING_ROUTING_PATH
OUTGOING_HEARTBEAT_TS     = INCOMING_HEARTBEAT_TS
OUTGOING_REPLICAT    = INCOMING_REPLICAT
OUTGOING_HEARTBEAT_TS    = INCOMING_HEARTBEAT_TS
```

Additional Considerations:

Computing lags as the heartbeat flows through the system relies on the clocks of the source and target systems to be set up correctly. It is possible that the lag can be negative if the target system is ahead of the source system. The lag is shown as a negative number so that you are aware of their clock discrepancy and can take actions to fix it.

The timestamp that flows through the system is in UTC. There is no time zone associated with the timestamp so when viewing the heartbeat tables, the lag can be viewed quickly even if different components are in different time zones. You can write any view you want on top of the underlying tables; UTC is recommended.

All the heartbeat entries are written to the trail in UTF-8.

The outgoing and incoming paths together uniquely determine a row. Meaning that if you have two rows with same outgoing path and a different incoming path, then it is considered two unique entries.

Heartbeat Table Details

The `GG_HEARTBEAT` table displays timestamp information of the end-to-end replication time and the timing information at the different components primary and secondary Extract and Replicat.

In a unidirectional environment, only the target database contains information about the replication lag. That is the time when a record is generated at the source database and becomes visible to clients at the target database.

> **Note:**
>
> The automatic heartbeat tables don't populate the `OUTGOING_%` columns with data, when both the source and remote databases have the same name. To change the database name, use the utility `DBNEWID`. For details, see the `DBNEWID` Utility.

Column	Data Type	Description
LOCAL_DATABASE	VARCHAR2	Local database where the replication time from the remote database is measured.
HEARTBEAT_TIMESTAMP	TIMESTAMP(6)	The point in time when a timestamp is generated at the remote database.
REMOTE_DATABASE	VARCHAR2	Remote database where the timestamp is generated
INCOMING_EXTRACT	VARCHAR2	Name of the primary Extract (capture) at the remote database
INCOMING_ROUTING_PATH	VARCHAR2	Name of the secondary Extract (pump) at the remote database
INCOMING_REPLICAT	VARCHAR2	Name of the Replicat on the local database.
INCOMING_HEARTBEAT_TS	TIMESTAMP(6)	Final timestamp when the information is inserted into the GG_HEARTBEAT table at the local database.
INCOMING_EXTRACT_TS	TIMESTAMP(6)	Timestamp of the generated timestamp is processed by the primary Extract at the remote database.
INCOMING_ROUTING_TS	TIMESTAMP(6)	Timestamp of the generated timestamp is processed by the secondary Extract at the remote database.
INCOMING_REPLICAT_TS	TIMESTAMP(6)	Timestamp of the generated timestamp is processed by Replicat at the local database.
OUTGOING_EXTRACT	VARCHAR2	Bidirectional/N-way replication: Name of the primary Extract on the local database.

Column	Data Type	Description
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	
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.

Column	Data Type	Description
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.
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.

Column	Data Type	Description
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.

Column	Data Type	Description
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.
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_TIM E.
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	
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.

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.
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 all other supported databases, you must create background jobs to update the heartbeat timestamp using the database specific scheduler functionality.

See ADD HEARTBEATTABLE, ALTER HEARTBEATTABLE for details on updating the heartbeat table.

Purge the Heartbeat History Tables

The heartbeat history table is purged periodically using a job. The default interval is 30 days and this interval can be specified or overridden using a command line inteface such as Admin Client or the Administration Service web interface.

For Oracle Database, the database job is created automatically.

For all other supported databases, you must create background jobs to purge the heartbeat history table using the database specific scheduler functionality.

Best Practice

Oracle recommends that you:

- Use the same heartbeat frequency on all the databases to makes diagnosis easier.
- Adjust the retention period if space is an issue.

- Retain the default heartbeat table frequency; the frequency set to be 30 to 60 seconds gives the best results for most workloads.
- Use lag history statistics to collect lag and age information.

Using the Automatic Heartbeat Commands

You can use the heartbeat table commands to control the Oracle GoldenGate automatic heartbeat functionality as follows.

Command	Description
`ADD HEARTBEATTABLE`	Creates the heartbeat tables required for automatic heartbeat functionality including the `LOGBSN` columns.
`ALTER HEARTBEATTABLE`	Alters existing heartbeat objects.
`ALTER HEARTBEATTABLE UPGRADE`	Alters the heartbeat tables to add the `LOGBSN` columns to the heartbeat tables. This is optional.
`DELETE HEARTBEATTABLE`	Deletes existing heartbeat objects.
`DELETE HEARTBEATENTRY`	Deletes entries in the heartbeat table.
`INFO HEARTBEATTABLE`	Displays heartbeat table information.

Db2 z/OS: Interpret Statistics for Update Operations

The actual number of DML operations that are executed on the Db2 database might not match the number of extracted DML operations that are reported by Oracle GoldenGate. Db2 does not log update statements if they do not physically change a row, so Oracle GoldenGate cannot detect them or include them in statistics.

Monitoring 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.

Using the Error Log

Use the Oracle GoldenGate error log to view:

- a history of commands
- Oracle GoldenGate processes that started and stopped
- processing that was performed
- errors that occurred
- informational and warning messages

Because the error log shows events as they occurred in sequence, it is a good tool for detecting the cause (or causes) of an error. For example, you might discover that:

- someone stopped a process
- a process failed to make a TCP/IP or database connection
- a process could not open a file

To view the error log, use any of the following:

- Standard shell command to view the `ggserr.log` file within the root Oracle GoldenGate directory
- Oracle GoldenGate Director or Oracle GoldenGate Monitor
- `VIEW GGSEVT` command.

You can control the `ggserr.log` file behavior to:

- Roll over the file when it reaches a maximum size, which is the default to avoid disk space issues.
- All messages are appended to the file by all processes without regard to disk space.
- Disable the file.
- Route messages to another destination, such as the system log.

This behavior is controlled and described in the `ogg-ggserr.xml` file in one of the following locations:

Microservices Architecture
`$OGG_HOME/etc/conf/logging/`

Using 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.

Topics:

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 proces 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 the 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 12-1 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

Table 12-1 (Cont.) Current Extract and Aged Reports

Permissions	X	Date	Report
-rw-rw-rw-	1 ggs ggs	3184 Sep 27 14:10	TCUST1.rpt
-rw-rw-rw-	1 ggs ggs	2655 Sep 27 14:06	TCUST2.rpt
-rw-rw-rw-	1 ggs ggs	2655 Sep 27 14:04	TCUST3.rpt
-rw-rw-rw-	1 ggs ggs	2744 Sep 27 13:56	TCUST4.rpt
-rw-rw-rw-	1 ggs ggs	3571 Aug 29 14:27	TCUST5.rpt

Reconcile the Time Differences

To account for time differences between source and target systems, use the `TCPSOURCETIMER | NOTCPSOURCETIMER` parameter in the Extract parameter file. This parameter adjusts the timestamps of replicated records for reporting purposes, making it easier to interpret synchronization lag.

Tuning

Learn about tuning the performance of Oracle GoldenGate.

Tuning the Performance of Oracle GoldenGate

See Tuning the Performance of Oracle GoldenGate in the *Administering Oracle GoldenGate* guide.

13
Autonomous Database

This section provides details about configuring Oracle GoldenGate with Oracle Autonomous Database, and using Extract and Replicat processes with Autonomous Database instances.

About Capturing and Replicating Data Using Autonomous Databases

Oracle GoldenGate can be used to replicate data into Oracle Autonomous Database from any certified source platform and replicate data from Oracle Autonomous Database to any certified target platform.

Use Case: When Using Oracle GoldenGate with Autonomous Databases

Using Oracle GoldenGate in the Oracle Autonomous Database can be configured to support the following scenarios:

- **Scalable Active-Active architecture**: Synchronize changes made across two or more databases to scale out workloads, provide increase resilience and near instantaneous failover across multiple data centers or regions.
- **Real-Time Data Warehouse**: Provide continuous, real-time capture and delivery of changed data between Oracle Autonomous Database systems.
- **Big Data Integration**: With Oracle GoldenGate for Big Data you can replicate data from the Oracle Autonomous Database to provide real-time streaming integration to all platforms supported by Big Data targets.
- **Real-Time Streaming Analytics**: Oracle GoldenGate integrates seamlessly with Oracle Stream Analytics to enable users to identify events of interest by executing queries against event streams in real time. It allows creating custom operational dashboards that provide real-time monitoring, transform streaming data, or raise alerts based on stream analysis.
- **Hybrid Replication**: Oracle GoldenGate replicates data from the Oracle Autonomous Database instance back to on-premise or to another cloud database or platform.

> **Note:**
>
> Oracle GoldenGate cannot be used to extract from the Always Free Autonomous Database due to lack of a supplemental logging feature in the database.

See Always Free Autonomous Database for details.

Details of Support When Using Oracle GoldenGate with Autonomous Databases

Review the supported data types and limitations before replicating data or from an Oracle Autonomous Database instance.

Oracle GoldenGate is supported for any type of Oracle Autonomous Database.

Oracle GoldenGate Replicat Limitations for Autonomous Databases

These are the limitations of Oracle GoldenGate when replicating to or from the Oracle Autonomous Database.

Supported Replicats
To replicate data into an Oracle Autonomous Database you must use Parallel Replicat (integrated or non-integrated mode) or Integrated Replicat.

Data Type Limitations for DDL and DML Replication

See the section Non-Supported Oracle Data Types.

Also see Data Types in the *Autonomous Database on Dedicated Exadata Infrastructure Documentation* and Data Types in the *Using Oracle Autonomous Database Serverless* guide.

DDL replication is supported depending on the restrictions in the Autonomous Databases.

Details of Support for Archived Log Retention

The two types of Autonomous Databases, Oracle Autonomous Database Serverless and Oracle Autonomous Database on Dedicated Exadata Infrastructure have different log retention behavior.

- Oracle Autonomous Database Serverless: Archived log files are kept in Fast Recovery Area (FRA) for up to 48 hours. After that, it is purged and the archived log files are moved to NFS mount storage, which is accessible by logminer. Three copies are created. The logminer should be able to access any of the copies. This is transparent to Oracle GoldenGate Extract. After it reaches 7 days, the NFS mounted copy is permanently removed. The Extract abends with the `archived log unavailable` error if the required archived log file is older than 7 days.
- Oracle Autonomous Database on Dedicated Exadata Infrastructure: When Oracle Autonomous Data Guard or Oracle GoldenGate is enabled, archived log files are kept in Fast Recovery Area (FRA) for up to 7 days. After that, the files are purged. There is no NFS mount location available for logminer to access archived log files that are older than 7 days. The Extract abends with the `archived log unavailable` error if the required archived log file is older than 7 days.

> **Note:**
>
> If the database instance is closed for more than 15 minutes, then the retention time is set back to 3 days. This implies that retention of archived log files is confirmed only for 3 days, regardless of whether the database instance is closed. The files are retained for 7 days only if the database instance is not closed.

Configure Extract to Capture from an Autonomous Database

Oracle Autonomous Database has a tight integration with Oracle GoldenGate. There are a number of differences when setting up Extract for an Autonomous database instance compared to a traditional Oracle Database.

Oracle Autonomous Database security has been enhanced to ensure that Extract is only able to capture changes from the specific tenant it connected to. 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:**

Run the `ALTER USER` command to unlock the `ggadmin` user and set the password for it. See Creating Users with Autonomous Database with Client-Side Tools.

This `ALTER USER` command must be run by the `admin` account user for Autonomous Databases.

```
ALTER USER ggadmin IDENTIFIED BY PASSWORD ACCOUNT UNLOCK;
```

Prerequisites for Configuring Oracle GoldenGate Extract to Capture from Autonomous Databases

Prior to configuring and starting the Extract process to capture from the Autonomous Database, make sure that the following requirements are met:

- Oracle Autonomous Database environment is provisioned and running.
- Autonomous Database-level supplemental logging should be enabled by the `ADMIN` or `GGADMIN`.

Configuring Autonomous Database Supplemental Logging for Extract

To add minimal supplemental logging to your Autonomous Database instance, log into the instance as `GGADMIN` or `ADMIN` account and execute the following commands:

```
ALTER PLUGGABLE DATABASE ADD SUPPLEMENTAL LOG DATA;
```

To `DROP` Autonomous Database-level supplemental logging incase you decide to stop capturing from that database instance:

```
ALTER PLUGGABLE DATABASE DROP SUPPLEMENTAL
        LOG DATA;
```

You can verify that the Autonomous Database-level supplemental logging is configured properly by issuing this SQL statement:

```
SELECT MINIMAL FROM dba_supplemental_logging;
```

The output for this statement is:

```
MINIMAL
-------
YES
```

The `MINIMAL` column will be `YES` if supplemental logging has been correctly set for this Autonomous Database instance.

Configure Extract to Capture from an Autonomous Database

Following are the steps to configure an Extract to capture from an Oracle Autonomous Database :

1. Install Oracle GoldenGate for your Oracle Autonomous Database instance.
2. Create a deployment for the Oracle GoldenGate environment. This is the deployment where the Extract that captures data from the Oracle Autonomous Database instance will be created. See Add a Deployment.
3. Obtain Oracle Autonomous Database Client Credentials.

   To establish connection to your Oracle Autonomous Database instance, download the client credentials file. To download client credentials, you can use the Oracle Cloud Infrastructure Console or Database Actions Launchpad. See Downloading Client Credentials (Wallets).

> **Note:**
>
> If you do not have administrator access to the Oracle Autonomous Database, you should ask your service administrator to download and provide the credentials files to you.

The following steps use the **Database Actions Launchpad** to download the client credentials.

a. Log in to your Oracle Autonomous Database account.

b. From the **Database Instance** page, click **Database Actions**. This launches the Database Actions Launchpad. The Launchpad attempts to log you into the database as ADMIN. If that is not successful, you will be prompted for your database ADMIN username and password.

c. On the **Database Actions** Launchpad, under **Administration**, click **Download Client Credentials (Wallets)**.

d. Enter a password to secure your Client Credentials zip file and click **Download**.

> **Note:**
>
> The password you provide when you download the wallet protects the downloaded Client Credentials wallet.

e. Save the credentials `zip` file to your local system.

The credentials `zip` file contains the following files:

- `cwallet.sso`
- `ewallet.p12`
- `keystore.jks`
- `ojdbc.properties`
- `sqlnet.ora`
- `tnsnames.ora`
- `truststore.jks`
- `ewallet.pem`
- `README.txt`

Refer and update (if required) the `sqlnet.ora` and `tnsnames.ora` files while configuring Oracle GoldenGate to work with the Autonomous Database instance.

4. Configure the server where Oracle GoldenGate is running to connect to the Autonomous Database instance.

a. Log in to the server where Oracle GoldenGate was installed.

b. Transfer the credentials `zip` file that you downloaded from Oracle Autonomous database instance to the Oracle GoldenGate server.

c. In the Oracle GoldenGate server, unzip the credentials file into a new directory, for example: `/u02/data/adwc_credentials`. This is your key directory.

d. To configure the connection details, open your `tnsnames.ora` file from the Oracle client location in the Oracle GoldenGate instance.

e. Use the connection string with the `LOW` consumer group *`dbname`*`_low`, for example, `graphdb1_low`, and move it to your local `tnsnames.ora` file.

See *Local Naming Parameters in the tnsnames.ora File* chapter in the *Oracle Database Net Services Reference* guide.

> **Note:**
>
> The `tnsnames.ora` file provided with the credentials file contains three database service names identifiable as:
>
> *ADWC_Database_Name*`_low`
> *ADWC_Database_Name*`_medium`
> *ADWC_Database_Name*`_high`
>
> Oracle recommends that you use *ADWC_Database_Name*`_low` with Oracle GoldenGate. See Predefined Database Service Names for Autonomous Database in the *Using Oracle Autonomous Database Serverless* guide or Predefined Database Service Names for Autonomous Databases for Oracle Autonomous Database on Dedicated Exadata Infrastructure.

f. Edit the `tnsnames.ora` file in the Oracle GoldenGate instance to include the connection details available in the `tnsnames.ora` file in your key directory (the directory where you unzipped the credentials `zip` file downloaded from the Autonomous Database.

Sample Connection String

```
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
```

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 exmaple:

```
ADD TRANDATA HR.EMP
```

or

```
ADD SCHEMATRANDATA HR
```

See Prerequisites for Configuring Oracle GoldenGate Extract to Capture from Autonomous Databases.

9. Add heartbeat table.

```
ADD HEARTBEATTABLE
```

10. Add and configure an Extract to capture from the Oracle Autonomous Database instance. See Add a Primary Extract for steps to create an Extract.

Oracle GoldenGate Extract is designed to work with the Oracle Autonomous Database instance to ensure that it only captures from a specific database instance. This means that the database instance name is not needed for any `TABLE` or `MAP` statements.

The following example creates an Extract (required for capturing from an Oracle Autonomous Database) called `exte`, and instructs it to begin now.

```
ADD EXTRACT exte, INTEGRATED TRANLOG, BEGIN NOW
```

To capture specific tables, use the two part object names.. For example, to capture from the table `HR.EMP`, in your Oracle Autonomous Database instance, use this entry in the Extract parameter file.

```
TABLE HR.EMP;
```

If you want to replicate `HR.EMP` into `COUNTRY.EMPLOYEE`, then your map statement would look like this:

```
MAP HR.EMP, TARGET COUNTRY.EMPLOYEE;
```

11. Register Extract with the Oracle Autonomous Database instance. For example, to register an Extract named `exte`, use the following command:

```
REGISTER EXTRACT exte DATABASE
```

12. You can now start your Extract and perform data replication to the Oracle Autonomous Database instance. Here's an example:

```
START EXTRACT exte
```

This completes the process of configuring an Extract for Oracle Autonomous Database and you can use it like any other Extract process.

Configure Replicat to Apply to an Oracle Autonomous Database

You can replicate into the Autonomous Database from any source database or platform that is certified by Oracle GoldenGate.

Prerequisites for Configuring Oracle GoldenGate Replicat to an Autonomous Database

You should have the following details available with you:

- Your source database with Oracle GoldenGate Extract processes configured and writing trails to where the Replicat is running to apply data to the Autonomous Database target.
- Oracle Autonomous Database is environment provisioned and running.

To deliver data to the Autonomous Database instance using Oracle GoldenGate, perform the following tasks:
Topics:

Configure Oracle GoldenGate for an Autonomous Database

Here are the steps to complete the configuration tasks:

> **Note:**
>
> Instructions are based on the assumption that the source environment is already configured. Learn the steps required to configure replication into the Autonomous 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 Database Serverless 21c. Oracle GoldenGate Microservices Architecture 21c and higher support Autonomous Database capture using Marketplace for Oracle Autonomous Database Serverless.
2. Create a deployment for your Oracle GoldenGate environment. This is the deployment where the Replicat that applies data into the Autonomous 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 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 Autonomous Database. It will be used in Replicat to allow Oracle GoldenGate to connect to the Autonomous 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 Database instance, you need to download the client credentials files. There are two ways to download the client credentials files: the Oracle Cloud Infrastructure Console or Database Actions Launchpad.

For details, see Downloading Client Credentials (Wallets).

> **Note:**
>
> If you do not have administrator access to the Oracle Autonomous Database, you should ask your service administrator to download and provide the credentials files to you.

The following steps use the **Database Actions Launchpad** to download the client credentials files.

1. Log in to your Autonomous Database account.
2. From the **Database Instance** page, click **Database Actions**. This launches the Database Actions Launchpad. The Launchpad attempts to log you into the database as ADMIN. If that is not successful, you will be prompted for your database ADMIN username and password.
3. On the **Database Actions** Launchpad, under **Administration**, click **Download Client Credentials (Wallets)**.
4. Enter a password to secure your Client Credentials zip file and click **Download**.

> **Note:**
>
> The password you provide when you download the wallet protects the downloaded Client Credentials wallet.

5. Save the credentials `zip` file to your local system. The credentials `zip` file contains the following files:
   - `cwallet.sso`
   - `ewallet.p12`
   - `keystore.jks`
   - `ojdbc.properties`
   - `sqlnet.ora`
   - `tnsnames.ora`
   - `truststore.jks`
   - `ewallet.pem`
   - `README.txt`

   Refer and update (if required) the `sqlnet.ora` and `tnsnames.ora` files while configuring Oracle GoldenGate to work with the Oracle Autonomous Database instance.

Configure Replicat to Apply to an Autonomous Database

This section assumes that the source environment is already configured and provides the steps required to establish replication in the Oracle Autonomous Database environment.

In the Oracle GoldenGate instance, you need to complete the following:

1. Follow the steps given in Prerequisites for Configuring Oracle GoldenGate Replicat to an Autonomous Database.
2. Follow the steps given in Configure Oracle GoldenGate for an Autonomous Database.
3. Follow the steps given in Obtain the Autonomous Database Client Credentials.
4. Log in to the server where Oracle GoldenGate was installed.
5. Transfer the credentials `zip` file that you downloaded from Oracle Autonomous Database to your Oracle GoldenGate instance.
6. In the Oracle GoldenGate instance, unzip the credentials file into a new directory `/u02/data/adwc_credentials`. This is your key directory.

7. To configure the connection details, open your `tnsnames.ora` file from the Oracle client location in the Oracle GoldenGate instance.

```
cd /u02/data/adwc_credentials
ls
tnsnames.ora
```

8. Edit the `tnsnames.ora` file in the Oracle GoldenGate instance to include the connection details available in the `tnsnames.ora` file in your key directory (the directory where you unzipped the credentials `zip` file downloaded from Oracle Autonomous Database).

```
Sample Connection String
graphdb1_low = (description=
                (retry_count=20)(retry_delay=3)(address=(protocol=tcps)
(port=1522)(host=adb-preprod.us-phoenix-1.oraclecloud.com))

(connect_data=(service_name=okd2ybgcz4mjx94_graphdb1_low.adb.oraclecloud.co
m))
                (security=(ssl_server_cert_dn="CN=adwc-preprod.uscom-
east-1.oraclecloud.com,OU=Oracle BMCS US,O=Oracle Corporation,L=Redwood
City,ST=California,C=US")))
```

If Replicat becomes unresponsive due to a network timeout or connection lost, then you can add the following into the connection profile in the `tnsnames.ora` file:

```
(DESCRIPTION =  (RECV_TIMEOUT=120)   (ADDRESS_LIST =
      (LOAD_BALANCE=off)(FAILOVER=on)(CONNECT_TIMEOUT=3)(RETRY_COUNT=3)
      (ADDRESS = (PROTOCOL = TCP)(HOST = adb-preprod.us-
phoenix-1.oraclecloud.com)(PORT = 1522))
```

> **Note:**
>
> The `tnsnames.ora` file provided with the credentials file contains three database service names identifiable as:
>
> *ADWC_Database_Name*_low
> *ADWC_Database_Name*_medium
> *ADWC_Database_Name*_high
>
> For Oracle GoldenGate replication, use *ADWC_Database_Name*_low.

9. To configure the wallet, create a `sqlnet.ora` file in the Oracle client location in the Oracle GoldenGate instance.

```
cd /u02/data/oci/network/admin
ls
sqlnet.ora tnsnames.ora
```

10. Edit this `sqlnet.ora` file to include your key directory.

```
WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA =
(DIRECTORY="/u02/data/adwc_credentials")))
SSL_SERVER_DN_MATCH=yes
```

11. Use the Admin Client to log in to the Oracle GoldenGate deployment.
12. Create a credential to store the `GGADMIN` user and password for the Replicat to use. For example:

```
ADD CREDENTIALSTORE ALTER CREDENTIALSTORE ADD USER
ggadmin@databasename_low PASSWORD complex_password alias adb_alias
```

13. Add and configure a Replicat to deliver to Oracle Autonomous Database. When creating the Replicat, use the alias created in the previous step. For setting up your Replicat and other processes, see Add a Replicat.

    The following example creates a Replicat (required to replicat to an Oracle Autonomous 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 Oracle Autonomous Database.

14. You can now start your Replicat and perform data replication to the Autonomous Database. Here's an example:

```
START REPLICAT rauto
```

> **Note:**
>
> Oracle Autonomous Database times out and disconnects the Replicat when it is idle for more than 60 minutes. When Replicat tries to apply changes (when it gets new changes) after being idle, it encounters a database error and abends. Oracle recommends that you configure Oracle GoldenGate with the `AUTORESTART` profile using managed processes (Microservices Architecture) to avoid having to manually restart a Replicat when it times out.

14
Upgrade

This section provides instructions for upgrading Oracle GoldenGate Microservices Architecture for Oracle database.

Topics:

Obtaining the Oracle GoldenGate Distribution

To obtain Oracle GoldenGate, follow these steps:

1. Go to edelivery: edelivery.oracle.com

   Also see MOS note 1645495.1 and 2193391.1 for more information.

   To access Oracle Technology Network, go to https://www.oracle.com/middleware/technologies/goldengate.html
2. Find the Oracle GoldenGate 21c release and download the ZIP file onto your system.

For more information about locating and downloading Oracle Fusion Middleware products, see the Oracle Fusion Middleware Download, Installation, and Configuration Readme Files on Oracle Technology Network.

Prerequisites

Learn about 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 `ETROLLOVER` 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.

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 Microservice 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.

Considerations for Upgrading Service Manager and other 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. There are no issues having a Service Manager running on the highest version and having deployments with lower versions.

After completing the upgrade, run the `UPGRADE HEARTBEATTABLE` command 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.

Topics:

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 integrated Extract version is upgraded.

Because the `TIMEZONE` datatype is managed differently with Oracle GoldenGate 21c, you may need to run the `ALTER REPLICAT` *`extseqno`* command to synchronize with newer trail files after consuming the old trail file written by Extract version 1.

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.

Because the `TIMEZONE` datatype is managed differently with Oracle GoldenGate 21c, you may need to run the `ALTER REPLICAT extseqno` command to synchronize with newer trail files after consuming the old trail file written by the Extract.

Upgrading Oracle GoldenGate Microservices – GUI Based

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

Follow these steps to obtain the Oracle GoldenGate installation software and set up the directories for upgrade.

1. Download the latest Oracle GoldenGate Microservices 21c software from the Oracle Technology Network or eDelivery.
2. Move the Oracle GoldenGate 21c MA software to a staging folder and unzip it.

   For Linux, use the following example:

   ```
   $ mv /home/user/fbo_ggs_Linux_x64_Oracle_services_shiphome.zip /tmp
   $ cd /tmp$ unzip fbo_ggs_Linux_x64_Oracle_services_shiphome.zip
   ```

3. Upload the Oracle GoldenGate Microservices 21c software to a staging location on the server where a previous release of Oracle GoldenGate Microservices exists.
4. Save the changes and return to the Service Manager Overview page.
5. Select the **Action** dropdown for the deployment and select **Restart**.
6. Log in to the Administration Service and click Action button in the Replicat section.
7. Click the Parameter File tab and change the value of the `BATCHSQL` parameter to double the value of `BATCHESPERQUEUE`. You must do this before starting Replicat. For example: `BATCHSQL BATCHESPERQUEUE 40000000`.
8. Log back into the Adminstration Service and start Extract and Replicat.

At this point, you should have a new Oracle GoldenGate 21c 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:

1. Log into Service Manager from the URL: *https://hostname:servicemanager_port*
2. Select the **ServiceManager** link in the **Deployments** section of the Service Manager Overview page.
3. Click the pencil icon next to the Deployment Detail section to open the dialog box for editing the GoldenGate home path.
4. Update the GoldenGate Home path with the full path to the new Oracle GoldenGate home.
5. Click **Apply**.
6. Use the **Action** dropdown to restart the Service Manager.

Upgrade the 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. Verify the current location of Oracle GoldenGate home directory from Service Manager.
   a. Login to the Service Manager: `http://hostname:servicemanager_port`
   b. Click the link to the deployment name in the **Deployments** section on the Service Manager Overview page. The deployment details are displayed.
3. Edit and update the the deployment with the location of the new Oracle GoldenGate Home directory.
   a. Click the pencil next to Service Manager Deployment Detail to edit the Oracle GoldenGate Home directory on the **Details** tab.
   b. Update the Oracle GoldenGate Home path with the complete path to the new Oracle GoldenGate home directory.
   c. Click **Apply**.
   d. Confirm that the Oracle GoldenGate Home path has been updated.
   e. Select the link for the **Administration Service** in the Deployment section.
   f. Log in and stop any Extracts and Replicats. Close the Administration Service page and return to the Service Manager page.
4. Return to the Deployment Detail page of the deployment and then select the **Configuration** tab to modify the settings for the environment variables. With the new Unified Build in Oracle GoldenGate 21c, the environment variables for `ORACLE_HOME`, `LD_LIBRARY_PATH`, and `TNS_ADMIN` need to be adjusted to the Oracle Database Client software within Oracle GoldenGate. Set the environment variables as:
   - `ORACLE_HOME` = `$OGG_HOME/lib/instantclient`
   - `LD_LIBRARY_PATH` = `$OGG_HOME/lib:$OGG_HOME/lib/instantclient`
   - `TNS_ADMIN` = *Location of tnsnames.ora and sqlnet.ora*
   - `JAVA_HOME` = `$OGG_HOME/jdk`
5. Save the changes and return to the Service Manager Overview page.
6. Save the changes and return to the Service Manager Overview page.
7. Select the **Action** dropdown for the deployment and select **Restart**.
8. Log back into the Adminstration Service and start any Extract and Replicats.

Upgrading Oracle GoldenGate Microservices Using REST APIs

Learn how to upgrade Oracle GoldenGate Microservices to Oracle GoldenGate Microservices 21c using REST APIs.

Follow these steps to obtain the Oracle GoldenGate installation software and set up the directories for upgrade.

1. Download the latest Oracle GoldenGate Microservices 21c software from the Oracle Technology Network or eDelivery.
2. Move the Oracle GoldenGate 21c MA software to a staging folder and unzip it.

   For Linux, use the following example:

   ```
   $ mv /home/user/fbo_ggs_Linux_x64_Oracle_services_shiphome.zip /tmp
   $ cd /tmp$ unzip fbo_ggs_Linux_x64_Oracle_services_shiphome.zip
   ```

3. Upload the Oracle GoldenGate Microservices 21c software to a staging location on the server where a previous release of Oracle GoldenGate Microservices exists.
4. Save the changes and return to the Service Manager home page.
5. Select the **Action** dropdown for the deployment and select **Restart**.
6. Log in to the Administration Service and click Action button in the Replicat section.
7. Click the Parameter File tab and change the value of the `BATCHSQL` parameter to double the value of `BATCHESPERQUEUE`. You must do this before starting Replicat. For example: `BATCHSQL BATCHESPERQUEUE 40000000`
8. Log back into the Adminstration Service and start Extract and Replicat.

Upgrade a 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/21c/gghome_1`.

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

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. Change the environment variables for the deployment, as shown in the following example:

```
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": "ORACLE_HOME" , "value": "/u01/app/oracle/
product/21c/gghome_1/lib/instantclient"}
                                    , {"name": "LD_LIBRARY_PATH" , "value":
"/u01/app/oracle/product/21c/gghome_1/lib/instantclient:/u01/app/oracle/
product/21c/gghome_1/lib"}
                                    , {"name": "JAVA_HOME" , "value":
"/u01/app/oracle/product/21c/gghome_1/jdk"}
                                    , {"name": "TNS_ADMIN" , "value":
"/u01/app/oracle/network/admin"} ] }'
```

3. 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"}'
```

4. Start all Extracts and Replicats.

When the Service Manager or deployment restarts, the upgrade is complete.

15

Appendix

Learn about additional details required for supporting Oracle GoldenGate on different databases.

Using the LogDump Utility to Access Trail File Records

Oracle GoldenGate trail information is required for troubleshooting and technical support. Use the Logdump utility to view the Oracle GoldenGate trail records.

Trail Recovery Mode

By default, Extract operates in append mode, where if there is a process failure, a recovery marker is written to the trail and Extract appends recovery data to the file so that a history of all prior data is retained for recovery purposes.

In append mode, the Extract initialization determines the identity of the last complete transaction that was written to the trail at startup time. With that information, Extract ends recovery when the commit record for that transaction is encountered in the data source; then it begins new data capture with the next committed transaction that qualifies for extraction and begins appending the new data to the trail. A Replicat starts reading again from that recovery point.

Overwrite mode is another version of Extract recovery that was used in versions of Oracle GoldenGate prior to version 10.0. In these versions, Extract overwrites the existing transaction data in the trail after the last write-checkpoint position, instead of appending the new data. The first transaction that is written is the first one that qualifies for extraction after the last read checkpoint position in the data source.

If the version of Oracle GoldenGate on the target is older than version 10, Extract will automatically revert to overwrite mode to support backward compatibility. This behavior can be controlled manually with the `RECOVERYOPTIONS` parameter.

Trail Record Format

Each change record written by Oracle GoldenGate to a trail or Extract file includes a header area, a data area, and possibly a user token area. The record header contains information about the transaction environment, and the data area contains the actual data values that were extracted.

The token area contains information that is specified by Oracle GoldenGate users for use in column mapping and conversion.

Oracle GoldenGate trail files are unstructured. You can view Oracle GoldenGate records with the Logdump utility provided with the Oracle GoldenGate software. For more information, see 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.

Topics:

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. Depracated 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.

Topics:

Partition Name Record in Trail File Header

Each DML record in the trail file header can contain an index to a partition name record (PNR). Because the full partition name can be long, a PNR is created in each trail file for the first time the partition is written. Each PNR, contains the partition name and partition object ID.

For primary Extract, PNR is generated only for partition matching and included by `PARTITION` and `PARTITIONEXCLUDE` parameters. DML records from these partitions have an index to the table definition record and another index to the partition name record. DML records from all other tables such as non-partitioned tables or partitioned tables not matching or excluded by the `PARTITION` or `PARTITIONEXCLUDE` parameters, only have an index to the table definition record as done today. For the Distribution Service, the PNR is written if source trail record contains a PNR index.

Viewing the Partition Name and PNR Index in Logdump

Use the Logdump utility to display the partition name record and the DML containing the PNR index.

Here's an example that shows capturing the display in a file:

```
$ logdump > output.txt <<EOF
ghdr on
detail data
open ./dirdat/tr000000000
n 200
EOF
```

The output displays the PNR and the DML with the PNR index values, as shown in the following example:

```
HDR-IND    :     E  (X45)         PARTITION  :     .  (XFF80)
UNDOFLAG   :     .  (X00)         BEFOREAFTER:     A  (X41)
RECLENGTH  :     0  (X0000)       IO TIME    : 2019/01/17 16:48:01.129.045
IOTYPE     :   170  (XAA)         ORIGNODE   :     4  (X04)
TRANSIND   :     .  (X03)         FORMATTYPE :     R  (X52)
SYSKEYLEN  :     0  (X00)         INCOMPLETE :     .  (X00)
TDR/PNR IDX:   (001, 002)       AUDITPOS   : 13287580
CONTINUED  :     N  (X00)         RECCOUNT   :     1  (X01)

2019/01/17 16:48:01.129.045 METADATA             LEN 0 RBA 3425
PARTITION NAME: P1  PARTITION ID: 75,234  FLAGS: X00000001

_____________________________________________________________________
HDR-IND    :     E  (X45)         PARTITION  :     .  (XFF8C)
UNDOFLAG   :     .  (X00)         BEFOREAFTER:     A  (X41)
RECLENGTH  :    18  (X0012)      IO TIME    : 2019/01/17 16:47:58.000.000
IOTYPE     :     5  (X05)        ORIGNODE   :   255  (XFF)
TRANSIND   :     .  (X00)         FORMATTYPE :     R  (X52)
SYSKEYLEN  :     0  (X00)         INCOMPLETE :     .  (X00)
AUDITRBA   :         15           AUDITPOS   : 13287580
CONTINUED  :     N  (X00)        RECCOUNT   :     1  (X01)

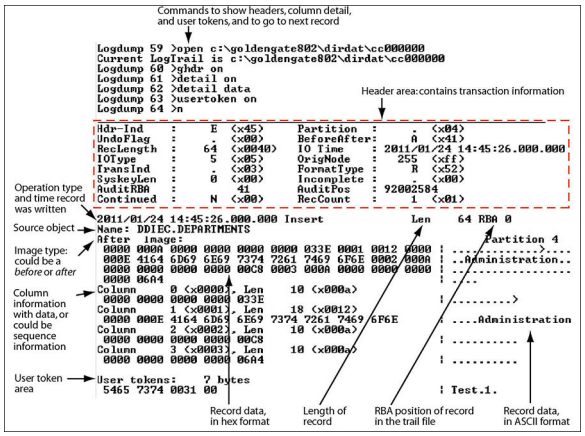
2019/01/17 16:47:58.000.000 INSERT               LEN    18 RBA 3486
NAME: TKGGU1.T1  (PARTITION: P1, TDR/PNR INDEX: 1/2)
AFTER  IMAGE:                                          PARTITION X8C   G  B
```

```
 0000 0500 0000 0100 3101 0005 0000 0001 0031      | ........1........1
COLUMN     0 (X0000), LEN     5 (X0005)
 0000 0100 31                                      | ....1
COLUMN     1 (X0001), LEN     5 (X0005)
 0000 0100 31                                      | ....1
```

Example of an Oracle GoldenGate Record

The following illustrates an Oracle GoldenGate record as viewed with Logdump. The first portion (the list of fields) is the header and the second portion is the data area. The record looks similar to this on all platforms supported by Oracle GoldenGate.

Figure 15-1 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.

Field	Description
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.
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

Field	Description
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

Using Header Data

The data area of the Oracle GoldenGate trail record contains the following:

- The time that the change was written to the Oracle GoldenGate file
- The type of database operation
- The length of the record
- The relative byte address within the trail file
- The table name
- The data changes in hex format

The following explains the differences in record image formats used by Oracle GoldenGate on Windows, UNIX, Linux, and NonStop systems.

Topics:

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).

- `colum_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 `COMPRESSDELETES` 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 `FieldComp` format (see below) and the update is compressed.	All
11-UpdateComp	A record/row in `TMF AuditComp` format was updated. In this format, only the changed bytes are present. A 4-byte descriptor in the format of `2-byte_offset2-byte_length` precedes each data fragment. The byte offset is the ordinal index of the column within the source table. The length is the length of the data.	NSK TMF
12-FileAlter	An attribute of a database file was altered.	NSK
13-FileCreate	A database file was created.	NSK
14-FilePurge	A database file was deleted.	NSK
15-FieldComp	A row in a SQL table was updated. In this format, only the changed bytes are present. Before images of unchanged columns are not logged by the database. A 4-byte descriptor in the format of `2-byte_offset2-byte_length` precedes each data fragment. The byte offset is the ordinal index of the column within the source table. The length is the length of the data. A partition indicator of 4 in the record header indicates `FieldComp` format.	All

Type	Description	Platform
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
115 and 117 (GGS)KeyFieldComp(32)	A primary key was updated. The Oracle GoldenGate record contains the before image of the key and the after image of the key and the row. The data is in `FieldComp` format (compressed), meaning that before images of unchanged columns are not logged by the database.	Windows and UNIX
116-LargeObject 116-LOB	Identifies a `RAW`, `BLOB`, `CLOB`, or `LOB` column. Data of this type is stored across multiple records.	Windows and UNIX
132-(GGS) SequenceOp	Identifies an operation on a sequence.	Windows and UNIX

Type	Description	Platform
134-UNIFIED UPDATE 135-UNIFIED PKUPDATE	Identifies a unified trail record that contains both before and after values in the same record. The before image in a `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-GGSComment	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 /orarac/oradata/racq/redo01.log
 2011-01-01 14:16:45 Thread 1, Seqno 47, RBA 68748800
Log Read Checkpoint File /orarac/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: /orarac/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: /orarac/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: /orarac/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: /orarac/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: /orarac/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: /orarac/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/eh000000000
 First Record RBA 3702915
Current Checkpoint Detail:
 Read Checkpoint #1
 GGS Log Trail
 Startup Checkpoint(starting position in data source):
 Sequence #: 0
 RBA: 3702915
 Timestamp: Not Available
 Extract Trail: ./dirdat/eh
 Current Checkpoint (position of last record read in the data source):
 Sequence #: 0
 RBA: 3702915
 Timestamp: Not Available
 Extract Trail: ./dirdat/eh
 Header:
 Version = 2
 Record Source = A
 Type = 1
 # Input Checkpoints = 1
 # Output Checkpoints =
 File Information:
 Block Size = 2048
 Max Blocks = 100
 Record Length = 2048
 Current Offset = 0
 Configuration:
 Data Source = 0
 Transaction Integrity = -1
 Task Type = 0
 Status:
 Start Time = 2011-01-12 13:10:13
 Last Update Time = 2011-01-12 21:23:31
 Stop Status = A
 Last Result = 400
```

Supported Character Sets

Here's a list of character sets that Oracle GoldenGate supports when converting data from source to target.

The identifiers that are shown should be used for Oracle GoldenGate parameters or commands when a character set must be specified, instead of the actual character set name. Currently Oracle GoldenGate does not provide a facility to specify the database-specific character set.

Supported Character Sets - Oracle

Table 15-1 Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
al32utf8	Unicode 9.0 Universal Character Set (UCS), UTF-8 encoding scheme
ar8ados710t	Arabic MS-DOS 710 8-bit Latin/Arabic
ar8ados710	Arabic MS-DOS 710 Server 8-bit Latin/Arabic
ar8ados720t	Arabic MS-DOS 720 8-bit Latin/Arabic
ar8ados720	Arabic MS-DOS 720 Server 8-bit Latin/Arabic
ar8aptec715t	APTEC 715 8-bit Latin/Arabic
ar8aptec715	APTEC 715 Server 8-bit Latin/Arabic
ar8arabicmacs	Mac Server 8-bit Latin/Arabic
ar8arabicmact	Mac 8-bit Latin/Arabic
ar8arabicmac	Mac Client 8-bit Latin/Arabic
ar8asmo708plus	ASMO 708 Plus 8-bit Latin/Arabic
ar8asmo8x	ASMO Extended 708 8-bit Latin/Arabic
ar8ebcdic420s	EBCDIC Code Page 420 Server 8-bit Latin/Arabic
ar8ebcdicx	EBCDIC XBASIC Server 8-bit Latin/Arabic
ar8hparabic8t	HP 8-bit Latin/Arabic
ar8iso8859p6	ISO 8859-6 Latin/Arabic
ar8mswin1256	MS Windows Code Page 1256 8-Bit Latin/Arabic
ar8mussad768t	Mussa'd Alarabi/2 768 8-bit Latin/Arabic
ar8mussad768	Mussa'd Alarabi/2 768 Server 8-bit Latin/Arabic
ar8nafitha711t	Nafitha International 711 Server 8-bit Latin/Arabic
ar8nafitha711	Nafitha Enhanced 711 Server 8-bit Latin/Arabic
ar8nafitha721t	Nafitha International 721 8-bit Latin/Arabic
ar8nafitha721	Nafitha International 721 Server 8-bit Latin/Arabic
ar8sakhr706	SAKHR 706 Server 8-bit Latin/Arabic
ar8sakhr707t	SAKHR 707 8-bit Latin/Arabic
ar8sakhr707	SAKHR 707 Server 8-bit Latin/Arabic
ar8xbasic	XBASIC 8-bit Latin/Arabic

Table 15-1 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
az8iso8859p9e	ISO 8859-9 Azerbaijani
bg8mswin	MS Windows 8-bit Bulgarian Cyrillic
bg8pc437s	IBM-PC Code Page 437 8-bit (Bulgarian Modification)
blt8cp921	Latvian Standard LVS8-92(1) Windows/Unix 8-bit Baltic
blt8ebcdic1112s	EBCDIC Code Page 1112 8-bit Server Baltic Multilingual
blt8ebcdic1112	EBCDIC Code Page 1112 8-bit Baltic Multilingual
blt8iso8859p13	ISO 8859-13 Baltic
blt8mswin1257	MS Windows Code Page 1257 8-bit Baltic
blt8pc775	IBM-PC Code Page 775 8-bit Baltic
bn8bscii	Bangladesh National Code 8-bit BSCII
cdn8pc863	IBM-PC Code Page 863 8-bit Canadian French
ce8bs2000	Siemens EBCDIC.DF.04-2 8-bit Central European
cel8iso8859p14	ISO 8859-13 Celtic
ch7dec	DEC VT100 7-bit Swiss (German/French)
cl8bs2000	Siemens EBCDIC.EHC.LC 8-bit Latin/Cyrillic-1
cl8ebcdic1025c	EBCDIC Code Page 1025 Client 8-bit Cyrillic
cl8ebcdic1025r	EBCDIC Code Page 1025 Server 8-bit Cyrillic
cl8ebcdic1025s	EBCDIC Code Page 1025 Server 8-bit Cyrillic
cl8ebcdic1025	EBCDIC Code Page 1025 8-bit Cyrillic
cl8ebcdic1025x	EBCDIC Code Page 1025 (Modified) 8-bit Cyrillic
cl8ebcdic1158r	EBCDIC Code Page 1158 Server 8-bit Cyrillic
cl8ebcdic1158	EBCDIC Code Page 1158 8-bit Cyrillic
cl8iso8859p5	ISO 8859-5 Latin/Cyrillic
cl8isoir111	SOIR111 Cyrillic
cl8koi8r	RELCOM Internet Standard 8-bit Latin/Cyrillic
cl8koi8u	KOI8 Ukrainian Cyrillic
cl8maccyrillics	Mac Server 8-bit Latin/Cyrillic
cl8maccyrillic	Mac Client 8-bit Latin/Cyrillic
cl8mswin1251	MS Windows Code Page 1251 8-bit Latin/Cyrillic
d7dec	DEC VT100 7-bit German
d7siemens9780x	Siemens 97801/97808 7-bit German
d8bs2000	Siemens 9750-62 EBCDIC 8-bit German
d8ebcdic1141	EBCDIC Code Page 1141 8-bit Austrian German
d8ebcdic273	EBCDIC Code Page 273/1 8-bit Austrian German

Table 15-1 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
dk7siemens9780x	Siemens 97801/97808 7-bit Danish
dk8bs2000	Siemens 9750-62 EBCDIC 8-bit Danish
dk8ebcdic1142	EBCDIC Code Page 1142 8-bit Danish
dk8ebcdic277	EBCDIC Code Page 277/1 8-bit Danish
e7dec	DEC VT100 7-bit Spanish
e7siemens9780x	Siemens 97801/97808 7-bit Spanish
e8bs2000	Siemens 9750-62 EBCDIC 8-bit Spanish
ee8bs2000	Siemens EBCDIC.EHC.L2 8-bit East European
ee8ebcdic870c	EBCDIC Code Page 870 Client 8-bit East European
ee8ebcdic870s	EBCDIC Code Page 870 Server 8-bit East European
ee8ebcdic870	EBCDIC Code Page 870 8-bit East European
ee8iso8859p2	ISO 8859-2 East European
ee8macces	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 ASCI West European/Greek
eec8europa3	EEC EUROPA3 8-bit West European/Greek
el8dec	DEC 8-bit Latin/Greek
el8ebcdic423r	IBM EBCDIC Code Page 423 for RDBMS server-side
el8ebcdic875r	EBCDIC Code Page 875 Server 8-bit Greek
el8ebcdic875s	EBCDIC Code Page 875 Server 8-bit Greek
el8ebcdic875	EBCDIC Code Page 875 8-bit Greek
el8gcos7	Bull EBCDIC GCOS7 8-bit Greek
el8iso8859p7	ISO 8859-7 Latin/Greek
el8macgreeks	Mac Server 8-bit Greek
el8macgreek	Mac Client 8-bit Greek
el8mswin1253	MS Windows Code Page 1253 8-bit Latin/Greek
el8pc437s	IBM-PC Code Page 437 8-bit (Greek modification)
el8pc737	IBM-PC Code Page 737 8-bit Greek/Latin
el8pc851	IBM-PC Code Page 851 8-bit Greek/Latin
el8pc869	IBM-PC Code Page 869 8-bit Greek/Latin

Table 15-1 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
et8mswin923	MS Windows Code Page 923 8-bit Estonian
f7dec	DEC VT100 7-bit French
f7siemens9780x	Siemens 97801/97808 7-bit French
f8bs2000	Siemens 9750-62 EBCDIC 8-bit French
f8ebcdic1147	EBCDIC Code Page 1147 8-bit French
f8ebcdic297	EBCDIC Code Page 297 8-bit French
hu8abmod	Hungarian 8-bit Special AB Mod
hu8cwi2	Hungarian 8-bit CWI-2
i7dec	DEC VT100 7-bit Italian
i7siemens9780x	Siemens 97801/97808 7-bit Italian
i8ebcdic1144	EBCDIC Code Page 1144 8-bit Italian
i8ebcdic280	EBCDIC Code Page 280/1 8-bit Italian
in8iscii	Multiple-Script Indian Standard 8-bit Latin/Indian
is8macicelandics	Mac Server 8-bit Icelandic
is8macicelandic	Mac Client 8-bit Icelandic
is8pc861	IBM-PC Code Page 861 8-bit Icelandic
iw7is960	Israeli Standard 960 7-bit Latin/Hebrew
iw8ebcdic1086	EBCDIC Code Page 1086 8-bit Hebrew
iw8ebcdic424s	EBCDIC Code Page 424 Server 8-bit Latin/Hebrew
iw8ebcdic424	EBCDIC Code Page 424 8-bit Latin/Hebrew
iw8iso8859p8	ISO 8859-8 Latin/Hebrew
iw8machebrews	Mac Server 8-bit Hebrew
iw8machebrew	Mac Client 8-bit Hebrew
iw8mswin1255	MS Windows Code Page 1255 8-bit Latin/Hebrew
iw8pc1507	IBM-PC Code Page 1507/862 8-bit Latin/Hebrew
ja16dbcs	IBM EBCDIC 16-bit Japanese
ja16ebcdic930	IBM DBCS Code Page 290 16-bit Japanese
ja16euctilde	Same as ja16euc except for the way that the wave dash and the tilde are mapped to and from Unicode
ja16euc	EUC 24-bit Japanese
ja16eucyen	EUC 24-bit Japanese with '\' mapped to the Japanese yen character
ja16macsjis	Mac client Shift-JIS 16-bit Japanese
ja16sjistilde	Same as ja16sjis except for the way that the wave dash and the tilde are mapped to and from Unicode.
ja16sjis	Shift-JIS 16-bit Japanese

Table 15-1 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
ja16sjisyen	Shift-JIS 16-bit Japanese with '\' mapped to the Japanese yen character
ja16vms	JVMS 16-bit Japanese
ko16dbcs	IBM EBCDIC 16-bit Korean
ko16ksc5601	KSC5601 16-bit Korean
ko16ksccs	KSCCS 16-bit Korean
ko16mswin949	MS Windows Code Page 949 Korean
la8iso6937	ISO 6937 8-bit Coded Character Set for Text Communication
la8passport	German Government Printer 8-bit All-European Latin
lt8mswin921	MS Windows Code Page 921 8-bit Lithuanian
lt8pc772	IBM-PC Code Page 772 8-bit Lithuanian (Latin/Cyrillic)
lt8pc774	IBM-PC Code Page 774 8-bit Lithuanian (Latin)
lv8pc1117	IBM-PC Code Page 1117 8-bit Latvian
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

Table 15-1 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
th8tisebcdics	Thai Industrial Standard 620-2533 - EBCDIC Server 8-bit
th8tisebcdic	Thai Industrial Standard 620-2533 - EBCDIC 8-bit
tr7dec	DEC VT100 7-bit Turkish
tr8dec	DEC 8-bit Turkish
tr8ebcdic1026s	EBCDIC Code Page 1026 Server 8-bit Turkish
tr8ebcdic1026	EBCDIC Code Page 1026 8-bit Turkish
tr8macturkishs	Mac Server 8-bit Turkish
tr8macturkish	Mac Client 8-bit Turkish
tr8mswin1254	MS Windows Code Page 1254 8-bit Turkish
tr8pc857	IBM-PC Code Page 857 8-bit Turkish
us7ascii	ASCII 7-bit American
us8bs2000	Siemens 9750-62 EBCDIC 8-bit American
us8icl	ICL EBCDIC 8-bit American
us8pc437	IBM-PC Code Page 437 8-bit American
vn8mswin1258	MS Windows Code Page 1258 8-bit Vietnamese
vn8vn3	VN3 8-bit Vietnamese
we8bs2000e	Siemens EBCDIC.DF.04-F 8-bit West European with Euro symbol
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

Table 15-1 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
we8ebcdic500	EBCDIC Code Page 500 8-bit West European
we8ebcdic871	EBCDIC Code Page 871 8-bit Icelandic
we8ebcdic924	Latin 9 EBCDIC 924
we8gcos7	Bull EBCDIC GCOS7 8-bit West European
we8hp	HP LaserJet 8-bit West European
we8icl	ICL EBCDIC 8-bit West European
we8iso8859p15	ISO 8859-15 West European
we8iso8859p1	ISO 8859-1 West European
we8iso8859p9	ISO 8859-9 West European & Turkish
we8isoicluk	ICL special version ISO8859-1
we8macroman8s	Mac Server 8-bit Extended Roman8 West European
we8macroman8	Mac Client 8-bit Extended Roman8 West European
we8mswin1252	MS Windows Code Page 1252 8-bit West European
we8ncr4970	NCR 4970 8-bit West European
we8nextstep	NeXTSTEP PostScript 8-bit West European
we8pc850	IBM-PC Code Page 850 8-bit West European
we8pc858	IBM-PC Code Page 858 8-bit West European
we8pc860	IBM-PC Code Page 860 8-bit West European
we8roman8	HP Roman8 8-bit West European
yug7ascii	ASCII 7-bit Yugoslavian
zhs16cgb231280	CGB2312-80 16-bit Simplified Chinese
zhs16dbcs	IBM EBCDIC 16-bit Simplified Chinese
zhs16gbk	GBK 16-bit Simplified Chinese
zhs16maccgb231280	Mac client CGB2312-80 16-bit Simplified Chinese
zht16big5	BIG5 16-bit Traditional Chinese
zht16ccdc	HP CCDC 16-bit Traditional Chinese
zht16dbcs	IBM EBCDIC 16-bit Traditional Chinese
zht16dbt	Taiwan Taxation 16-bit Traditional Chinese
zht16hkscs31	MS Windows Code Page 950 with Hong Kong Supplementary Character Set HKSCS-2001 (character set conversion to and from Unicode is based on Unicode 3.1)
zht16hkscs	MS Windows Code Page 950 with Hong Kong Supplementary Character Set HKSCS-2001 (character set conversion to and from Unicode is based on Unicode 3.0)
zht16mswin950	MS Windows Code Page 950 Traditional Chinese
zht32euc	EUC 32-bit Traditional Chinese

Table 15-1 (Cont.) Supported Oracle Character Sets

Identifier to use in parameter files and commands	Character Set
zht32sops	SOPS 32-bit Traditional Chinese
zht32tris	TRIS 32-bit Traditional Chinese

Supported Character Sets - Non-Oracle

Identifier to use in parameter files and commands	Character set
UTF-8	ISO-10646 UTF-8, surrogate pairs are 4 bytes per character
UTF-16	ISO-10646 UTF-16
UTF-16BE	UTF-16 Big Endian
UTF-16LE	UTF-16 Little Endian
UTF-32	ISO-10646 UTF-32
UTF-32BE	UTF-32 Big Endian
UTF-32LE	UTF-32 Little Endian
CESU-8	Similar to UTF-8, correspond to UCS-2 and surrogate pairs are 6 bytes per character
US-ASCII	US-ASCII, ANSI X34-1986
windows-1250	Windows Central Europe
windows-1251	Windows Cyrillic
windows-1252	Windows Latin-1
windows-1253	Windows Greek
windows-1254	Windows Turkish
windows-1255	Windows Hebrew
windows-1256	Windows Arabic

Identifier to use in parameter files and commands	Character set
windows-1257	Windows Baltic
windows-1258	Windows Vietnam
windows-874	Windows Thai
cp437	DOS Latin-1
ibm-720	DOS Arabic
cp737	DOS Greek
cp775	DOS Baltic
cp850	DOS multilingual
cp851	DOS Greek-1
cp852	DOS Latin-2
cp855	DOS Cyrillic
cp856	DOS Cyrillic / IBM
cp857	DOS Turkish
cp858	DOS Multilingual with Euro
cp860	DOS Portuguese
cp861	DOS Icelandic
cp862	DOS Hebrew
cp863	DOS French
cp864	DOS Arabic
cp865	DOS Nordic
cp866	DOS Cyrillic / GOST 19768-87

Identifier to use in parameter files and commands	Character set
ibm-867	DOS Hebrew / IBM
cp868	DOS Urdu
cp869	DOS Greek-2
ISO-8859-1	ISO-8859-1 Latin-1/Western Europe
ISO-8859-2	ISO-8859-2 Latin-2/Eastern Europe
ISO-8859-3	ISO-8859-3 Latin-3/South Europe
ISO-8859-4	ISO-8859-4 Latin-4/North Europe
ISO-8859-5	ISO-8859-5 Latin/Cyrillic
ISO-8859-6	ISO-8859-6 Latin/Arabic
ISO-8859-7	ISO-8859-7 Latin/Greek
ISO-8859-8	ISO-8859-8 Latin/Hebrew
ISO-8859-9	ISO-8859-9 Latin-5/Turkish
ISO-8859-10	ISO-8859-10 Latin-6/Nordic
ISO-8859-11	ISO-8859-11 Latin/Thai
ISO-8859-13	ISO-8859-13 Latin-7/Baltic Rim
ISO-8859-14	ISO-8859-14 Latin-8/Celtic
ISO-8859-15	ISO-8859-15 Latin-9/Western Europe
IBM037	IBM 037-1/697-1 EBCDIC, Brazil, Canada, Netherlands, Portugal, US, and 037/1175 Traditional Chinese
IBM01140	IBM 1140-1/695-1 EBCDIC, Brazil, Canada, Netherlands, Portugal, US, and 1140/1175 Traditional Chinese
IBM273	IBM 273-1/697-1 EBCDIC, Austria, Germany
IBM01141	IBM 1141-1/695-1 EBCDIC, Austria, Germany

Identifier to use in parameter files and commands	Character set
IBM277	IBM 277-1/697-1 EBCDIC, Denmark, Norway
IBM01142	IBM 1142-1/695-1 EBCIDC, Denmark, Norway
IBM278	IBM 278-1/697-1 EBCDIC, Finland, Sweden
IBM01143	IBM 1143-1/695-1 EBCDIC, Finland, Sweden
IBM280	IBM 280-1/697-1 EBCDIC, Italy
IBM01144	IBM 1144-1/695-1 EBCDIC, Italy
IBM284	IBM 284-1/697-1 EBCDIC, Latin America, Spain
IBM01145	IBM 1145-1/695-1 EBCDIC, Latin America, Spain
IBM285	IBM 285-1/697-1 EBCDIC, United Kingdom
IBM01146	IBM 1146-1/695-1 EBCDIC, United Kingdom
IBM290	IBM 290 EBCDIC, Japan (Katakana) Extended
IBM297	IBM 297-1/697-1 EBCDIC, France
IBM01147	IBM 1147-1/695-1 EBCDIC, France
IBM420	IBM 420 EBCDIC, Arabic Bilingual
IBM424	IBM 424/941 EBCDIC, Israel (Hebrew - Bulletin Code)
IBM500	IBM 500-1/697-1 EBCDIC, International
IBM01148	IBM 1148-1/695-1 EBCDIC International
IBM870	IBM 870/959 EBCDIC, Latin-2 Multilingual
IBM871	IBM 871-1/697-1 EBCDIC Iceland
IBM918	IBM EBCDIC code page 918, Arabic 2
IBM1149	IBM 1149-1/695-1, EBCDIC Iceland

Identifier to use in parameter files and commands	Character set
IBM1047	IBM 1047/103 EBCDIC, Latin-1 (Open Systems)
ibm-803	IBM 803 EBCDIC, Israel (Hebrew - Old Code)
IBM875	IBM 875 EBCDIC, Greece
ibm-924	IBM 924-1/1353-1 EBCDIC International
ibm-1153	IBM 1153/1375 EBCDIC, Latin-2 Multilingual
ibm-1122	IBM 1122/1037 EBCDIC, Estonia
ibm-1157	IBM 1157/1391 EBCDIC, Estonia
ibm-1112	IBM 1112/1035 EBCDIC, Latvia, Lithuania
ibm-1156	IBM 1156/1393 EBCDIC, Latvia, Lithuania
ibm-4899	IBM EBCDIC code page 4899, Hebrew with Euro
ibm-12712	IBM 12712 EBCDIC, Hebrew (max set including Euro)
ibm-1097	IBM 1097 EBCDIC, Farsi
ibm-1018	IBM 1018 EBCDIC, Finland Sweden (ISO-7)
ibm-1132	IBM 1132 EBCDIC, Laos
ibm-1137	IBM EBCDIC code page 1137, Devanagari
ibm-1025	IBM 1025/1150 EBCDIC, Cyrillic
ibm-1154	IBM EBCDIC code page 1154, Cyrillic with Euro
IBM1026	IBM 1026/1152 EBCDIC, Latin-5 Turkey
ibm-1155	IBM EBCDIC code page 1155, Turkish with Euro
ibm-1123	IBM 1123 EBCDIC, Ukraine
ibm-1158	IBM EBCDIC code page 1158, Ukranian with Euro

Identifier to use in parameter files and commands	Character set
IBM838	IBM 838/1173 EBCDIC, Thai
ibm-1160	IBM EBCDIC code page 1160, Thai with Euro
ibm-1130	IBM 1130 EBCDIC, Vietnam
ibm-1164	IBM EBCDIC code page 1164, Vietnamese with Euro
ibm-4517	IBM EBCDIC code page 4517, Arabic French
ibm-4971	IBM EBCDIC code page 4971, Greek
ibm-9067	IBM EBCDIC code page 9067, Greek 2005
ibm-16804	IBM EBCDIC code page 16804, Arabic
KOI8-R	Russian and Cyrillic (KOI8-R)
KOI8-U	Ukranian (KOI8-U)
eucTH	EUC Thai
ibm-1162	Windows Thai with Euro
DEC-MCS	DEC Multilingual
hp-roman8	HP Latin-1 Roman8
ibm-901	IBM Baltic ISO-8 CCSID 901
ibm-902	IBM Estonia ISO-8 with Euro CCSID 902
ibm-916	IBM ISO8859-8 CCSID
ibm-922	IBM Estonia ISO-8 CCSID 922
ibm-1006	IBM Urdu ISO-8 CCSID 1006
ibm-1098	IBM Farsi PC CCSID 1098
ibm-1124	Ukranian ISO-8 CCSID 1124

Identifier to use in parameter files and commands	Character set
ibm-1125	Ukranian without Euro CCSID 1125
ibm-1129	IBM Vietnamese without Euro CCSID 1129
ibm-1131	IBM Belarusi CCSID 1131
ibm-1133	IBM Lao CCSID 1133
ibm-4909	IBM Greek Latin ASCII CCSID 4909
JIS_X201	JIS X201 Japanese
windows-932	Windows Japanese
windows-936	Windows Simplified Chinese
ibm-942	IBM Windows Japanese
windows-949	Windows Korean
windows-950	Windows Traditional Chinese
eucjis	EUC Japanese
EUC-JP	IBM/MS EUC Japanese
EUC-CN	EUC Simplified Chinese, GBK
EUC-KR	EUC Korean
EUC-TW	EUC Traditional Chinese
ibm-930	IBM 930/5026 Japanese
ibm-933	IBM 933 Korean
ibm-935	IBM 935 Simplified Chinese
ibm-937	IBM 937 Traditional Chinese
ibm-939	IBM 939/5035 Japanese

Identifier to use in parameter files and commands	Character set
ibm-1364	IBM 1364 Korean
ibm-1371	IBM 1371 Traditional Chinese
ibm-1388	IBM 1388 Simplified Chinese
ibm-1390	IBM 1390 Japanese
ibm-1399	IBM 1399 Japanese
ibm-5123	IBM CCSID 5123 Japanese
ibm-8482	IBM CCSID 8482 Japanese
ibm-13218	IBM CCSID 13218 Japanese
ibm-16684	IBM CCSID 16684 Japanese
shiftjis	Japanese Shift JIS, Tilde 0x8160 mapped to U+301C
gb18030	GB-18030
GB2312	GB-2312-1980
GBK	GBK
HZ	HZ GB2312
Ibm-1381	IBM CCSID 1381 Simplified Chinese
Big5	Big5, Traditional Chinese
Big5-HKSCS	Big5, HongKong ext.
Big5-HKSCS2001	Big5, HongKong ext. HKSCS-2001
ibm-950	IBM Big5, CCSID 950
ibm-949	CCSID 949 Korean
ibm-949C	IBM CCSID 949 Korean, has backslash

Identifier to use in parameter files and commands	Character set
ibm-971	IBM CCSID 971 Korean EUC, KSC5601 1989
x-IBM1363	IBM CCSID 1363, Korean

Supported Locales

Here's a list of the locales that are supported by Oracle GoldenGate. The locale is used when comparing case-insensitive object names.

```
af
af_NA
af_ZA
am
am_ET
ar
ar_AE
ar_BH
ar_DZ
ar_EG
ar_IQ
ar_JO
ar_KW
ar_LB
ar_LY
ar_MA
ar_OM
ar_QA
ar_SA
ar_SD
ar_SY
ar_TN
ar_YE
as
as_IN
az
az_Cyrl
az_Cyrl_AZ
az_Latn
az_Latn_AZ
be
be_BY
bg
bg_BG
bn
```

```
bn_BD
bn_IN
ca
ca_ES
cs
cs_CZ
cy
cy_GB
da
da_DK
de
de_AT
de_BE
de_CH
de_DE
de_LI
de_LU
el
el_CY
el_GR
en
en_AU
en_BE
en_BW
en_BZ
en_CA
en_GB
en_HK
en_IE
en_IN
en_JM
en_MH
en_MT
en_NA
en_NZ
en_PH
en_PK
en_SG
en_TT
en_US
en_US_POSIX
en_VI
en_ZA
en_ZW
eo
es
es_AR
es_BO
es_CL
es_CO
```

```
es_CR
es_DO
es_EC
es_ES
es_GT
es_HN
es_MX
es_NI
es_PA
es_PE
es_PR
es_PY
es_SV
es_US
es_UY
es_VE
et
et_EE
eu
eu_ES
fa
fa_AF
fa_IR
fi
fi_FI
fo
fo_FO
fr
fr_BE
fr_CA
fr_CH
fr_FR
fr_LU
fr_MC
ga
ga_IE
gl
gl_ES
gu
gu_IN
gv
gv_GB
haw
haw_US
he
he_IL
hi
hi_IN
hr
hr_HR
```

```
hu
hu_HU
hy
hy_AM
hy_AM_REVISED
id
id_ID
is
is_IS
it
it_CH
it_IT
ja
ja_JP
ka
ka_GE
kk
kk_KZ
kl
kl_GL
km
km_KH
kn
kn_IN
ko
ko_KR
kok
kok_IN
kw
kw_GB
lt
lt_LT
lv
lv_LV
mk
mk_MK
ml
ml_IN
mr
mr_IN
ms
ms_BN
ms_MY
mt
mt_MT
nb
nb_NO
nl
nl_BE
nl_NL
```

```
nn
nn_NO
om
om_ET
om_KE
or
or_IN
pa
pa_Guru
pa_Guru_IN
pl
pl_PL
ps
ps_AF
pt
pt_BR
pt_PT
ro
ro_RO
ru
ru_RU
ru_UA
sk
sk_SK
sl
sl_SI
so
so_DJ
so_ET
so_KE
so_SO
sq
sq_AL
sr
sr_Cyrl
sr_Cyrl_BA
sr_Cyrl_ME
sr_Cyrl_RS
sr_Latn
sr_Latn_BA
sr_Latn_ME
sr_Latn_RS
sv
sv_FI
sv_SE
sw
sw_KE
sw_TZ
ta
ta_IN
```

```
te
te_IN
th
th_TH
ti
ti_ER
ti_ET
tr
tr_TR
uk
uk_UA
ur
ur_IN
ur_PK
uz
uz_Arab
uz_Arab_AF
uz_Cyrl
uz_Cyrl_UZ
uz_Latn
uz_Latn_UZ
vi
vi_VN
zh
zh_Hans
zh_Hans_CN
zh_Hans_SG
zh_Hant
zh_Hant_HK
zh_Hant_MO
zh_Hant_TW
```

Commit Sequence Number (CSN)

When working with Oracle GoldenGate, you might need to refer to a Commit Sequence Number (CSN). A CSN is an identifier that Oracle GoldenGate constructs to identify a transaction for the purpose of maintaining transactional consistency and data integrity. It uniquely identifies a point in time in which a transaction commits to the database.

The CSN can be required to position Extract in the transaction log, to reposition Replicat in the trail, or for other purposes. It is returned by some conversion functions and is included in reports and certain command output.

A CSN is a monotonically increasing identifier generated by Oracle GoldenGate that uniquely identifies a point in time when a transaction commits to the database. It purpose is to ensure transactional consistency and data integrity as transactions are replicated from source to target. Each kind of database management system generates some kind of unique serial number of its own at the completion of each transaction, which uniquely identifies the commit of that transaction. For example, the Oracle RDBMS generates a System Change Number, which is a monotonically increasing sequence number assigned to every event by Oracle RDBMS. The CSN captures this same identifying information and represents it internally as a series of bytes, but the CSN is processed in a platform-independent manner. A comparison of

any two CSN numbers, each of which is bound to a transaction-commit record in the same log stream, reliably indicates the order in which the two transactions completed.

The CSN is cross-checked with the transaction ID (displayed as XID in Oracle GoldenGate informational output). The XID-CSN combination uniquely identifies a transaction even in cases where there are multiple transactions that commit at the same time, and thus have the same CSN. For example, this can happen in an Oracle RAC environment, where there is parallelism and high transaction concurrency.

The CSN value is stored as a token in any trail record that identifies the commit of a transaction. This value can be retrieved with the `@GETENV` column conversion function and viewed with the Logdump utility.

Using the Commit Sequence Number

This appendix contains information about using the Oracle GoldenGate Commit Sequence Number (CSN) with Oracle and non-Oracle databases.

All database platforms except Oracle, Db2 LUW, and Db2 z/OS have fixed-length CSNs, which are padded with leading zeroes as required to fill the fixed length. CSNs that contain multiple fields can be padded within each field. For more information on CSN, see in Overview: Commit Sequence Number (CSN) in the *Oracle GoldenGate Microservices* guide.

MySQL does not create a transaction ID as part of its event data, so Oracle GoldenGate considers a unique transaction identifier to be a *combination* of the following:

- the log file number of the log file that contains the `START TRANSACTION` record for the transaction that is being identified
- the record offset of that record

Table 15-2 Oracle GoldenGate CSN Values Per Database

Database	CSN Value
Db2 for i	*sequence_number* Where: • sequence_number is the fixed-length, 20 digit, decimal-based Db2 for i journal sequence number. Example: `12345678901234567890`

Db2 LUW	`LRI`
	Where:
	For version 10.1 and later, *LRI* is a period-separated pair of numbers for the Db2 log record identifier.
	Example:
	`123455.34645`

Table 15-2 (Cont.) Oracle GoldenGate CSN Values Per Database

Database	CSN Value
Db2 z/OS	*LSN* **where**: • `LSN` is, up to 20 hexadecimal digit representation of the 10 byte `LSN` in the transaction log. **Note:** Oracle GoldenGate uses `LSN` to represent both the non-data sharing `LSN` and data sharing `LRSN` as the format is same. **Example:** `0x1A3367F6BA12289`
MySQL	*LogNum*`:`*LogPosition* **Where:** • *LogNum* is the the name of the log file that contains the `START TRANSACTION` record for the transaction that is being identified. • *LogPosition* is the event offset value of that record. Event offset values are stored in the record header section of a log record. For example, if the log number is 12 and the log position is 121, the CSN is: `000012:000000000000121`

MySQL (Group Replication)	`SeqNum:GTID`
	
	In the preceding syntax:
	• `SeqNum` is the Oracle GoldenGate sequence number.
	• `GTID` the MySQL global transaction identifier.
	For example, if the sequence number is `00000000000000000001` and the GTID is `f77024f9-f4e3-11eb-a052-0021f6e03f10:0000000000000010654`, then the CSN value is:
	
	`00000000000000000001:f77024f9-f4e3-11eb-a052-0021f6e03f10:0000000000000010654`

Table 15-2 (Cont.) Oracle GoldenGate CSN Values Per Database

Database	CSN Value
Oracle	*system_change_number* **Where:** • *system_change number* is the Oracle SCN value. **Example:** 6488359
SQL Server	Can be any of these, depending on how the database returns it: • Colon separated hex string (8:8:4) padded with leading zeroes and 0X prefix • Colon separated decimal string (10:10:5) padded with leading zeroes • Colon separated hex string with 0X prefix and without leading zeroes • Colon separated decimal string without leading zeroes • Decimal string **Where:** • The first value is the virtual log file number, the second is the segment number within the virtual log, and the third is the entry number. **Examples:** 0X00000d7e:0000036b:01bd 0000003454:0000000875:00445 0Xd7e:36b:1bd 3454:875:445 3454000000087500445

Connecting Microservices and Classic Architectures

This topic lists the steps to establish a connection between Oracle GoldenGate Microservices and Classic architectures.

Connect Oracle GoldenGate Classic Architecture to Microservices Architecture

Oracle GoldenGate Classic Architecture uses the data pump Extract in Admin Client and GGSCI to connect to Microservices Architecture

> **Note:**
>
> Oracle GoldenGate Classic Architecture's pump Extract can only connect to an un-secured 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.

Options	Description
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, `.dirdat`. d. **Target trail file name**: Name of the target trail file, such as `ea`.
Generated Target URI	The location of the target trail file is displayed.
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.

Options	Description
Target Type	Select **Manager** as the target type. Alternatively, you can select **Collector** or **Receiver Service**. When connecting Microservices architecture with other Microservices architecture, select the **Receiver Service** option. When connecting Microservices architecture with Classic architecture, select either the **Manager** or **Collector** option. If you select the **Collector** option, you need to start a static collector beforehand on the Classic architecture and use that static collector port as the value of the **Target Manager Port** field.
Begin	Select the **Position in Log** option from the drop-down list.
Source Sequence Number	Enter the sequence value of the source trail.
Source RBA Offset	Enter the value of the RBA offset of the source trail if you want the path to start reading from a specific RBA.

4. Click `Create Path` or `Create and Run`, as required. Select `Cancel` if you need to get out of the **Add Path** page without adding a path.

After the path is created, you'll be able to see the new path in the Distribution Service home page.