

Oracle® Fusion Middleware

Using Oracle GoldenGate for Oracle Database



19c (19.1.0.0)

E98068-08

September 2023

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

Oracle Fusion Middleware Using Oracle GoldenGate for Oracle Database, 19c (19.1.0.0)

E98068-08

Copyright © 1995, 2023, Oracle and/or its affiliates.

Primary Author: Oracle Corp.

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

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

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

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

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

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

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

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience	xiii
Documentation Accessibility	xiii
Related Information	xiii
Conventions	xiii

1 Understanding What's Supported

Details of Support for Oracle Data Types and Objects	1-1
Non-Supported Oracle Data Types	1-5
Details of Support for Objects and Operations in Oracle DML	1-6
Multitenant Container Database	1-6
Tables, Views, and Materialized Views	1-6
Limitations of Support for Regular Tables	1-7
Limitations of Support for Views	1-9
Limitations of Support for Materialized Views	1-9
Limitations of Support for Clustered Tables	1-9
Sequences and Identity Columns	1-9
Limitations of Support for Sequences	1-9
Non-supported Objects and Operations in Oracle DML	1-10
Details of Support for Objects and Operations in Oracle DDL	1-10
Supported Objects and Operations in Oracle DDL	1-10
Non-supported Objects and Operations in Oracle DDL	1-12
Excluded Objects	1-12
Other Non-supported DDL	1-14
Integrating Oracle GoldenGate into a Cluster	1-14
General Requirements in a Cluster	1-15
Adding Oracle GoldenGate as a Windows Cluster Resource	1-15

2 Preparing the Database for Oracle GoldenGate

Configuring Connections for Integrated Processes	2-1
Configuring Logging Properties	2-2

Enabling Minimum Database-level Supplemental Logging	2-4
Enabling Schema-level Supplemental Logging	2-5
Enabling Table-level Supplemental Logging	2-7
Enabling Oracle GoldenGate in the Database	2-8
Setting Flashback Query	2-8
Managing Server Resources	2-10
Ensuring Row Uniqueness in Source and Target Tables	2-10

3 Establishing Oracle GoldenGate Credentials

Assigning Credentials to Oracle GoldenGate	3-1
Oracle GoldenGate Users (Database)	3-1
Granting the Appropriate User Privileges	3-2
Securing the Oracle GoldenGate Credentials	3-5

4 Choosing Capture and Apply Modes

Overview of Oracle GoldenGate Capture and Apply Processes	4-1
Deciding Which Capture Method to Use	4-2
About Integrated Capture	4-2
Integrated Capture Deployment Options	4-4
About Classic Capture	4-4
Deciding Which Apply Method to Use	4-5
About Parallel Replicat	4-5
About Non-integrated Replicat	4-7
About the Integrated Replicat Mode	4-7
Benefits of Integrated Replicat	4-9
Integrated Replicat Requirements	4-10
Using Different Capture and Apply Modes Together	4-10
Switching to a Different Process Mode	4-11

5 Using Parallel Replicat

Parallel Replication Architecture	5-1
Basic Parameters for Parallel Replicat	5-2
Creating a Parallel Replicat	5-3

6 Configuring Capture in Integrated Mode

Prerequisites for Configuring Integrated Capture	6-1
What to Expect from these Instructions	6-2
Configuring the Primary Extract in Integrated Capture Mode	6-2

Configuring the Data Pump Extract	6-5
Next Steps	6-7

7 Configuring Oracle GoldenGate Apply

Prerequisites for Configuring Replicat	7-1
What to Expect from these Instructions	7-2
Creating a Checkpoint Table	7-2
Adding the Checkpoint Table to the Target Database	7-3
Specifying the Checkpoint Table in the Oracle GoldenGate Configuration	7-3
Disabling Default Asynchronous COMMIT to Checkpoint Table	7-3
Configuring Replicat	7-4
Next Steps	7-6

8 Additional Oracle GoldenGate Configuration Considerations

Installing Support for Oracle Sequences	8-1
Handling Special Data Types	8-3
Multibyte Character Types	8-3
Oracle Spatial Objects	8-4
TIMESTAMP	8-5
Large Objects (LOB)	8-5
XML	8-5
User Defined Types	8-6
Handling Other Database Properties	8-6
Controlling the Checkpoint Frequency	8-7
Excluding Replicat Transactions	8-7
Advanced Configuration Options for Oracle GoldenGate	8-8

9 Additional Configuration Steps For Using Nonintegrated Replicat

Disabling Triggers and Referential Cascade Constraints on Target Tables	9-1
---	-----

10 Configuring DDL Support

Prerequisites for Configuring DDL	10-2
Support for DDL Capture in Integrated Capture Mode	10-2
Support for DDL Capture in Classic Capture Mode	10-3
Overview of DDL Synchronization	10-3
Limitations of Oracle GoldenGate DDL Support	10-3
DDL Statement Length	10-4
Supported Topologies	10-4

Filtering, Mapping, and Transformation	10-4
Renames	10-5
Interactions Between Fetches from a Table and DDL	10-5
Comments in SQL	10-6
Compilation Errors	10-6
Interval Partitioning	10-6
DML or DDL Performed Inside a DDL Trigger	10-6
LogMiner Data Dictionary Maintenance	10-6
Configuration Guidelines for DDL Support	10-6
Database Privileges	10-7
Parallel Processing	10-7
Object Names	10-7
Data Definitions	10-7
Truncates	10-8
Initial Synchronization	10-8
Data Continuity After CREATE or RENAME	10-8
Understanding DDL Scopes	10-8
Mapped Scope	10-9
Unmapped Scope	10-10
Other Scope	10-11
Correctly Identifying Unqualified Object Names in DDL	10-11
Enabling DDL Support	10-12
Filtering DDL Replication	10-12
Filtering with PL/SQL Code	10-13
Filtering With Built-in Filter Rules	10-15
DDL_AUX.addRule() Function Definition	10-15
Parameters for DDL_AUX.addRule()	10-15
Valid DDL Components for DDL_AUX.addRule()	10-16
Examples of Rule-based Trigger Filtering	10-17
Dropping Filter Rules	10-17
Filtering with the DDL Parameter	10-18
Special Filter Cases	10-19
DDL EXCLUDE ALL	10-19
Implicit DDL	10-19
How Oracle GoldenGate Handles Derived Object Names	10-20
MAP Exists for Base Object, But Not Derived Object	10-20
MAP Exists for Base and Derived Objects	10-21
MAP Exists for Derived Object, But Not Base Object	10-22
New Tables as Derived Objects	10-22
CREATE TABLE AS SELECT	10-22
RENAME and ALTER TABLE RENAME	10-23

Disabling the Mapping of Derived Objects	10-23
Using DDL String Substitution	10-25
Controlling the Propagation of DDL to Support Different Topologies	10-25
Propagating DDL in Active-Active (Bidirectional) Configurations	10-26
Propagating DDL in a Cascading Configuration	10-28
Adding Supplemental Log Groups Automatically	10-28
Removing Comments from Replicated DDL	10-28
Replicating an IDENTIFIED BY Password	10-28
How DDL is Evaluated for Processing	10-29
Viewing DDL Report Information	10-31
Viewing DDL Reporting in Replicat	10-32
Viewing DDL Reporting in Extract	10-32
Statistics in the Process Reports	10-33
Tracing DDL Processing	10-34
Using Tools that Support Trigger-Based DDL Capture	10-34
Tracing the DDL Trigger	10-34
Viewing Metadata in the DDL History Table	10-34
Handling DDL Trigger Errors	10-34
Using Edition-Based Redefinition	10-35

11 Creating Process Groups

Prerequisites	11-1
Registering Extract with the Mining Database	11-2
Add the Primary Extract	11-3
Add the Local Trail	11-5
Add the Data Pump Extract Group	11-5
Add the Remote Trail	11-5
Add the Replicat Group	11-6

12 Instantiating Oracle GoldenGate Replication

Overview of the Instantiation Process	12-1
Prerequisites for Instantiation	12-2
Configuring and Adding Change Synchronization Groups	12-2
Disabling DDL Processing	12-2
Adding Collision Handling	12-2
Preparing the Target Tables	12-3
Configuring the Initial Load	12-3
Configuring a Load with an Oracle Data Pump	12-3
Configuring a Direct Bulk Load to SQL*Loader	12-4

Configuring a Load from an Input File to SQL*Loader	12-6
Performing the Target Instantiation	12-9
Performing Instantiation with Oracle Data Pump	12-9
Performing Instantiation with Direct Bulk Load to SQL*Loader	12-10
Performing Instantiation From an Input File to SQL*Loader	12-11
Monitoring and Controlling Processing After the Instantiation	12-12
Verifying Synchronization	12-13
Backing up the Oracle GoldenGate Environment	12-13

13 Managing the DDL Replication Environment

Disabling DDL Processing Temporarily	13-2
Enabling and Disabling the DDL Trigger	13-2
Maintaining the DDL Marker Table	13-2
Deleting the DDL Marker Table	13-3
Maintaining the DDL History Table	13-3
Deleting the DDL History Table	13-3
Purging the DDL Trace File	13-4
Applying Database Patches and Upgrades when DDL Support is Enabled	13-4
Apply Oracle GoldenGate Patches and Upgrades when DDL support is Enabled	13-5
Restoring an Existing DDL Environment to a Clean State	13-6
Removing the DDL Objects from the System	13-8

14 Automatic Conflict Detection and Resolution

About Automatic Conflict Detection and Resolution	14-1
Automatic Conflict Detection and Resolution	14-2
Requirements for Automatic Conflict Detection and Resolution	14-3
Latest Timestamp Conflict Detection and Resolution	14-3
Delta Conflict Detection and Resolution	14-4
Column Groups	14-6
Configuring Automatic Conflict Detection and Resolution	14-8
Configuring Latest Timestamp Conflict Detection and Resolution	14-8
Configuring Delta Conflict Detection and Resolution	14-10
Managing Automatic Conflict Detection and Resolution	14-10
Altering Conflict Detection and Resolution for a Table	14-11
Altering a Column Group	14-11
Purging Tombstone Rows	14-12
Removing Conflict Detection and Resolution From a Table	14-12
Removing a Column Group	14-12
Removing Delta Conflict Detection and Resolution	14-13

Monitoring Automatic Conflict Detection and Resolution	14-13
Displaying Information About the Tables Configured for Conflicts	14-14
Displaying Information About Conflict Resolution Columns	14-14
Displaying Information About Column Groups	14-16

15 Using Procedural Replication

About Procedural Replication	15-1
Procedural Replication Process Overview	15-2
Enabling Procedural Replication	15-3
Determining Whether Procedural Replication Is On	15-3
Enabling and Disabling Supplemental Logging	15-4
Filtering Features for Procedural Replication	15-4
Handling Procedural Replication Errors	15-6
Procedural Replication Pragma Options	15-6
Listing the Procedures Supported for Oracle GoldenGate Procedural Replication	15-38
Monitoring Oracle GoldenGate Procedural Replication	15-39

16 Configuring Oracle GoldenGate in a Multitenant Container Database

Capturing from Pluggable Databases	16-1
Applying to Pluggable Databases	16-1
Excluding Objects from the Configuration	16-2
Other Requirements for Multitenant Container Databases	16-2

17 Using Oracle GoldenGate with Autonomous Database

About Capturing and Replicating Data Using Autonomous Databases	17-1
Details of Support When Using Oracle GoldenGate with Autonomous Databases	17-2
Configuring Replicat to Apply to an Autonomous Database	17-3
Prerequisites for Configuring Oracle GoldenGate Replicat to an Autonomous Database	17-3
Configure Oracle GoldenGate Replicat for an Autonomous Database	17-3
Obtain the Autonomous Database Client Credentials	17-4
Configure Replicat to Apply to an Autonomous Database	17-5
Configuring Extract to Capture from an Autonomous Database	17-7
Establishing Oracle GoldenGate Credentials	17-8
Prerequisites for Configuring Oracle GoldenGate Extract to Capture from Autonomous Databases	17-8
Configure Extract to Capture from an Autonomous Database	17-9

A Optional Parameters for Integrated Modes

Additional Parameter Options for Integrated Capture	A-1
Additional Parameter Options for Integrated Replicat	A-2

B Configuring a Downstream Mining Database

Evaluating Capture Options for a Downstream Deployment	B-1
Preparing the Source Database for Downstream Deployment	B-1
Creating the Source User Account	B-2
Configuring Redo Transport from Source to Downstream Mining Database	B-2
Preparing the Downstream Mining Database	B-4
Creating the Downstream Mining User Account	B-4
Configuring the Mining Database to Archive Local Redo Log Files	B-4
Preparing a Downstream Mining Database for Real-time Capture	B-5
Create the Standby Redo Log Files	B-5
Configure the Database to Archive Standby Redo Log Files Locally	B-7

C Example Downstream Mining Configuration

Example 1: Capturing from One Source Database in Real-time Mode	C-1
Prepare the Mining Database to Archive its Local Redo	C-2
Prepare the Mining Database to Archive Redo Received in Standby Redo Logs from the Source Database	C-2
Prepare the Source Database to Send Redo to the Mining Database	C-2
Set up Integrated Capture (ext1) on DBMSCAP	C-3
Example 2: Capturing from Multiple Sources in Archive-log-only Mode	C-4
Prepare the Mining Database to Archive its Local Redo	C-4
Prepare the Mining Database to Archive Redo from the Source Database	C-5
Prepare the First Source Database to Send Redo to the Mining Database	C-5
Prepare the Second Source Database to Send Redo to the Mining Database	C-5
Set up Extracts at Downstream Mining Database	C-6
Example 3: Capturing from Multiple Sources with Mixed Real-time and Archive-log-only Mode	C-6
Prepare the Mining Database to Archive its Local Redo	C-7
Prepare the Mining Database to Accept Redo from the Source Databases	C-7
Prepare the First Source Database to Send Redo to the Mining Database	C-7
Prepare the Second Source Database to Send Redo to the Mining Database	C-8
Prepare the Third Source Database to Send Redo to the Mining Database	C-8
Set up Extracts at Downstream Mining Database	C-9
Set up Extract (ext1) to Capture Changes from Archived Logs Sent by DBMS1	C-9
Set up Extract (ext2) to Capture Changes from Archived Logs Sent by DBMS2	C-10

D Installing Trigger-Based DDL Capture

When to Use Trigger-based DDL Capture	D-1
Overview of the Objects that Support Trigger-based DDL Capture	D-2
Installing the DDL Objects	D-3

E Supporting Changes to XML Schemas

Supporting RegisterSchema	E-1
Supporting DeleteSchema	E-1
Supporting CopyEvolve	E-1

F Preparing DBFS for an Active-Active Configuration

Supported Operations and Prerequisites	F-1
Applying the Required Patch	F-2
Examples Used in these Procedures	F-2
Partitioning the DBFS Sequence Numbers	F-2
Configuring the DBFS file system	F-3
Mapping Local and Remote Peers Correctly	F-5

G Support for Classic Extract

Details of Support for Objects and Operations in Oracle DML	G-3
Limitations of Support for Index-Organized Tables	G-3
Limitations of Support for Clustered Tables	G-3
Non-supported Objects and Operations in Oracle DML (Classic)	G-3
Details of Support for Objects and Operations in Oracle DDL (Classic)	G-4

H Configuring Capture in Classic Mode

Prerequisites for Configuring Classic Capture	H-1
What to Expect from these Instructions	H-2
Configuring the Primary Extract in Classic Capture Mode	H-2
Configuring the Data Pump Extract	H-3
Next Steps	H-5

Additional Configuration Steps for Using Classic Capture

Configuring Oracle TDE Data in Classic Capture Mode	I-1
Overview of TDE Support in Classic Capture Mode	I-2
Requirements for Capturing TDE in Classic Capture Mode	I-2
Required Database Patches for TDE Support	I-3
Configuring Classic Capture for TDE Support	I-3
Agree on a Shared Secret that Meets Oracle Standards	I-3
Oracle DBA Tasks	I-3
Oracle Security Officer Tasks	I-4
Oracle GoldenGate Administrator Tasks	I-5
Recommendations for Maintaining Data Security after Decryption	I-6
Performing DDL while TDE Capture is Active	I-6
Rekeying after a Database Upgrade	I-6
Updating the Oracle Shared Secret in the Parameter File	I-6
Using Classic Capture in an Oracle RAC Environment	I-7
Mining ASM-stored Logs in Classic Capture Mode	I-8
Accessing the Transaction Logs in ASM	I-9
Reading Transaction Logs Through the RDBMS	I-9
ASM Direct Connection	I-9
Ensuring ASM Connectivity	I-10
Ensuring Data Availability for Classic Capture	I-10
Log Retention Requirements per Extract Recovery Mode	I-11
Log Retention Options	I-11
All Other Oracle Versions	I-12
Determining How Much Data to Retain	I-12
Purging Log Archives	I-12
Specifying the Archive Location	I-12
Mounting Logs that are Stored on Other Platforms	I-13
Configuring Classic Capture in Archived Log Only Mode	I-13
Limitations and Requirements for Using ALO Mode	I-13
Configuring Extract for ALO mode	I-14
Configuring Classic Capture in Oracle Active Data Guard Only Mode	I-15
Limitations and Requirements for Using ADG Mode	I-15
Configuring Classic Extract for ADG Mode	I-17
Migrating Classic Extract To and From an ADG Database	I-17
Handling Role Changes In an ADG Configuration	I-18
Avoiding Log-read Bottlenecks in Classic Capture	I-20

Preface

This guide helps you get started with using Oracle GoldenGate on Oracle Database.

Topics:

- [Audience](#)
- [Documentation Accessibility](#)
- [Related Information](#)
- [Conventions](#)

Audience

Using Oracle GoldenGate for Oracle Databases is intended for DBA and system administrators who are responsible for implementing Oracle GoldenGate and managing the Oracle database.

Documentation Accessibility

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

Access to Oracle Support

Oracle 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 Libraries are found at

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

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

[Oracle GoldenGate A-Team Chronicles](#)

Conventions

The following text conventions are used in this document:

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

1

Understanding What's Supported

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

Topics:

- [Details of Support for Oracle Data Types and Objects](#)
This topic describes data types, objects and operations that are supported by Oracle GoldenGate.
- [Details of Support for Objects and Operations in Oracle DML](#)
This section outlines the Oracle objects and operations that Oracle GoldenGate supports for the capture and replication of DML operations.
- [Details of Support for Objects and Operations in Oracle DDL](#)
This topic outlines the Oracle objects and operation types that Oracle GoldenGate supports for the capture and replication of DDL operations.
- [Integrating Oracle GoldenGate into a Cluster](#)
If you installed Oracle GoldenGate in a cluster, take the following steps to integrate Oracle GoldenGate within the cluster solution.

Details of Support for Oracle Data Types and Objects

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

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

- Support by Capturing from Redo
- Procedural Replication Support

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

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

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

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

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

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



Note:

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

Supported Capture from Redo:

- NUMBER, BINARY FLOAT, BINARY DOUBLE, and (logical) UROWID
- DATE and TIMESTAMP
- CHAR, VARCHAR2, LONG, NCHAR, and NVARCHAR2
- RAW, LONG RAW, CLOB, NCLOB, BLOB, SECUREFILE, BASICFILE, and BFILE (LOB size limited to 4GB)
- XML columns stored as CLOB, Binary and Object-Relational (OR)
- XMLType columns and XMLType tables stored as XML CLOB, XML Object Relational, and XML Binary
- UDTs (user-defined or abstract data types) on BYTE semantics with source database compatibility 12.0.0.0.0 or higher
- ANYDATA data type with source database compatibility 12.0.0.0.0 or higher

- Hierarchy-enabled tables are managed by the Oracle XML database repository with source database compatibility 12.2.0.0.0 or higher and enabled procedural replication
- REF types with source database compatibility 12.2.0.0.0 or higher
- DICOM with source database compatibility 12.0.0.0.0 or higher
- SDO_TOPO_GEOMETRY or SDO_GEORASTER with source database compatibility 12.2.0.0.0 or higher and enabled procedural replication
- Identity columns with source database compatibility 18.1.0.0.0 or higher
- SDO_RDF_TRIPLE_S with source database compatibility 19.1.0.0.0 or higher

Supported (Fetch from database)

SECUREFILE LOBs

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

UDTs that contain following data types:

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

Object tables contains the following attributes:

- Nested table
- SDO_TOP_GEOMETRY
- SDO_GEORASTER

Additional Considerations

- NUMBER can be up to the maximum size permitted by Oracle. The support of the range and precision for floating-point numbers depends on the host machine. In general, the precision is accurate to 16 significant digits, but you should review the database documentation to determine the expected approximations. Oracle GoldenGate rounds or truncates values that exceed the supported precision.
- Non-logical UROWID columns will be identified by Extract. A warning message is generated in the report file. The column information is not part of the trail record. All other supported datatypes of the record are part of the trail record and are replicated.
- TIMESTAMP WITH TIME ZONE as TZR (region ID) for initial loads, SQLEXEC or operations where the column can only be fetched from the database. In those cases, the region ID is converted to a time offset by the source database when the column is selected. Replicat applies the timestamp as date and time data into the target database with a time offset value.
- VARCHAR expansion from 4K to 32K (extended or long VARCHAR)
 - 32K long columns cannot be used as row identifiers:

- * Columns as part of a key or unique index
- * Columns in a `KEYCOLS` clause of the `TABLE` or `MAP` parameter.
- 32K long columns as resolution columns in a CDR (conflict resolution and detection)
- If an extended `VARCHAR` column is part of unique index or constraint, then direct path inserts to this table may cause Replicat 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;`
- Oracle GoldenGate does not support the filtering, column mapping, or manipulation of objects larger than 4K.
- `BFILE` column are replicating the locator. The file on the server file system outside of the database and is not replicated.
- Multi-byte character data: The source and target databases must be logically identical in terms of schema definition for the tables and sequences being replicated. Transformation, filtering, and other manipulation cannot be used.
- The character sets between the two databases must be one of the following:
 - Identical on the source and on the target
 - Equivalent, which is not the same character set but containing the same set of characters
 - Target is a superset of the source

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

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

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

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

Oracle recommends the `AL32UTF8` character set as the database character set when working with `XML` data. This ensures the correct conversion by Oracle GoldenGate from source to target. With DDL replication enabled, Oracle GoldenGate replicates the CTAS statement and allows it to select the data from

the underlying target tables. OIDs are preserved if `TRANSLOGOPTIONS GETCTASDML` parameter is set. For `XMLType` tables, the row object IDs must match between source and target.

- [Non-Supported Oracle Data Types](#)

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. Oracle GoldenGate treats XMLType data as an LOB.
 - VARCHAR2 (MAX) where the data is greater than 32KB
- Tables with LOB, UDT, XML, or XMLType column without one of the following:
 - Primary Key
 - Scalar columns with a unique constraint or unique indexTable where the combination of all scalar columns do not guarantee uniqueness are unsupported.
- Tables with the following XML characteristics:
 - Tables with a primary key constraint made up of XML attributes
 - XMLType tables with a primary key based on an object identifier (PKOID).
 - XMLType tables, where the row object identifiers (OID) do not match between source and target
 - XMLType tables created by an empty CTAS statement.
 - XML schema-based XMLType tables and columns where changes are made to the XML schema (XML schemas must be registered on source and target databases with the `dbms_xml` package).
 - The maximum length for the entire `SET` value of an update to an XMLType larger than 32K, including the new content plus other operators and XQuery bind values.
 - SQL*Loader direct-path insert for XML-Binary and XML-OR.
- Tables with following UDT characteristics:
 - UDTs that contain CFILE or OPAQUE (except of XMLType)
 - UDTs with CHAR and VARCHAR attributes that contain binary or unprintable characters
 - UDTs using the RMTTASK parameter
- UDTs and nested tables with following condition:
 - Nested table UDTs with CHAR, NVARCHAR2 or NCLOB attributes.
 - Nested tables with CLOB, BLOB, extended (32k) VARCHAR2 or RAW attributes in UDTs.

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

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

Details of Support for Objects and Operations in Oracle DML

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

Supported Objects and Operations in Oracle DML

Topics:

- [Multitenant Container Database](#)
- [Tables, Views, and Materialized Views](#)
- [Sequences and Identity Columns](#)
- [Non-supported Objects and Operations in Oracle DML](#)

Multitenant Container Database

Oracle GoldenGate captures from, and delivers to, a **multitenant container database**, see [Configuring Oracle GoldenGate in a Multitenant Container Database](#).

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`.

- [Limitations of Support for Regular Tables](#)
- [Limitations of Support for Views](#)
- [Limitations of Support for Materialized Views](#)
- [Limitations of Support for Clustered Tables](#)

Limitations of Support for Regular Tables

These limitations apply to Extract.

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

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

Limitations of Support for Views

These limitations apply to Extract.

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

Limitations of Support for Materialized Views

Materialized views are supported by Extract with the following limitations.

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

Limitations of Support for Clustered Tables

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

Sequences and Identity Columns

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

Limitations of Support for Sequences

These limitations apply to integrated and classic capture modes.

- 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 Coordinated 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, but not primary-key based `REFs` (`PKREFs`)
- Sequence values in an active-active bi-directional configuration
- Database Replay
- Tables created as `EXTERNAL`

Details of Support for Objects and Operations in Oracle DDL

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

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

- [Supported Objects and Operations in Oracle DDL](#)
- [Non-supported Objects and Operations in Oracle DDL](#)

Supported Objects and Operations in Oracle DDL

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

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



Note:

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

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

- All Oracle GoldenGate topology configurations are supported for Oracle DDL replication.

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

EBR does not support use of DDL triggers.
- Oracle GoldenGate supports DDL operations of up to 4 MB in size. Oracle GoldenGate measures the size of DDL statement in bytes, not in characters. This size limitation includes packages, procedures, and functions. The actual size limit of the DDL support is approximate, because the size not only includes the statement text, but also Oracle GoldenGate maintenance overhead that depends on the length of the object name, the DDL type, and other characteristics of keeping a DDL record internally.
- Oracle GoldenGate supports Global Temporary Tables (GTT) DDL operations to be visible to Extract so that they can be replicated. You must set the `DDLOPTIONS` parameter to enable this operation because it is not set by default.

- Oracle GoldenGate supports Integrated Dictionary for use with `NOUSERID` and `TRANLOGOPTIONS GETCTASDML`. This means that Extract will be obtaining object metadata from the LogMiner dictionary instead of the DDL trigger and without querying the dictionary objects. Oracle GoldenGate uses Integrated Dictionary automatically when the source database compatibility parameter is greater than or equal to 11.2.0.4 and Integrated Extract is used.

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

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

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

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

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

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

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

Non-supported Objects and Operations in Oracle DDL

These statements apply to integrated and classic capture modes.

- [Excluded Objects](#)
- [Other Non-supported 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
"OJVM SYS", // Java Policy SRO Schema
"OLAPSYS", // OLAP catalogs
"ORACLE_OCM", // Oracle Configuration Manager User
"ORDDATA", // Intermedia
"ORDPLUGINS", // Intermedia
"ORDSYS", // Intermedia
"OUTLN", // Outlines (Plan Stability)
"SI_INFORMTN_SCHEMA", // SQL/MM Still Image
"SPATIAL_CSW_ADMIN", // Spatial Catalog Services for Web
"SPATIAL_CSW_ADMIN_USR",
"SPATIAL_WFS_ADMIN", // Spatial Web Feature Service
"SPATIAL_WFS_ADMIN_USR",
"SYS",
"SYSBACKUP",
"SYSDG",
"SYSKM",
"SYSMAN", // Administrator OEM
"SYSTEM",
"TSMSYS", // Transparent Session Migration
"WKPROXY", // Ultrasearch
"WKSYS", // Ultrasearch
"WK_TEST",
"WMSYS", // Workspace Manager
"XDB", // XML DB
"XS$NULL",
"XTISYS", // Time Index

```

Special schemas:

```

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

```

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

```

"*.AQ$*", // advanced queues
"*.DR$*$*", // oracle text
"*.M*_*$*", // Spatial index

```

```
".MLOG$", // materialized views
".OGGQT$",
".OGG$", // AQ OGG queue table
".ET$", // Data Pump external tables
".RUPD$", // materialized views
".SYS_C*", // constraints
".MDR*_*$", // Spatial Sequence and Table
".SYS_IMPORT_TABLE*",
".CMP*_*$", // space management, rdbms >= 12.1
".DBMS_TABCOMP_TEMP_*$", // space management, rdbms < 12.1
".MDXT_*_*$" // Spatial extended statistics tables
```

Other Non-supported DDL

Oracle GoldenGate does not support the following:

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

Integrating Oracle GoldenGate into a Cluster

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

For more information about installing and using Oracle GoldenGate in a cluster, see the following white papers for Classic and Microservices architectures:

1. [Oracle GoldenGate Classic Architecture with Oracle Real Application Clusters Configuration Best Practices](#)
 2. [Oracle GoldenGate Microservices Architecture with Oracle Real Application Clusters Configuration Best Practices](#)
- [General Requirements in a Cluster](#)
 - [Adding Oracle GoldenGate as a Windows Cluster Resource](#)

General Requirements in a Cluster

1. Configure the Oracle Grid Infrastructure Bundled Agent (XAG) to automatically manage the GoldenGate processes on the cluster nodes. See [Configuring Oracle GoldenGate with Oracle Grid Infrastructure Bundled Agents \(XAG\)](#) to know more.

Using the XAG makes sure that the required cluster file system is mounted before the Oracle GoldenGate processes are started. If an application virtual IP (VIP) is used in the cluster the bundled agent will also ensure the VIP is started on the correct node.
2. Configure the Oracle GoldenGate Manager process with the `AUTOSTART` and `AUTORESTART` parameters so that Manager starts the replication processes automatically.
3. Mount the shared drive on one node only. This prevents processes from being started on another node. Use the same mount point on all nodes. If you are using the Oracle Grid Infrastructure Bundled Agent, the mounting of the required file systems are automatically carried out.
4. Ensure that all database instances in the cluster have the same `COMPATIBLE` parameter setting.
5. Configure Oracle GoldenGate as directed in this documentation.

Adding Oracle GoldenGate as a Windows Cluster Resource

When installing Oracle GoldenGate in a Windows cluster, follow these instructions to establish Oracle GoldenGate as a cluster resource and configure the Manager service correctly on all nodes.

- In the cluster administrator, add the Manager process to the group that contains the database instance to which Oracle GoldenGate will connect.
- Make sure all nodes on which Oracle GoldenGate will run are selected as possible owners of the resource.
- Make certain the Manager Windows service has the following dependencies (can be configured from the Services control panel):
 - The database resource
 - The disk resource that contains the Oracle GoldenGate directory
 - The disk resource that contains the database transaction log files
 - The disk resource that contains the database transaction log backup files

2

Preparing the Database for Oracle GoldenGate

Learn how to prepare your database for Oracle GoldenGate, including how to configure connections and logging, how to enable Oracle GoldenGate in your database, how to set the flashback query, and how to manage server resources.

Topics:

- [Configuring Connections for Integrated Processes](#)
If you will be using integrated capture and integrated Replicat, each requires a dedicated server connection in the `tnsnames.ora` file.
- [Configuring Logging Properties](#)
Oracle GoldenGate relies on the redo logs to capture the data that it needs to replicate source transactions. The Oracle redo logs on the source system must be configured properly before you start Oracle GoldenGate processing.
- [Enabling Oracle GoldenGate in the Database](#)
The database services required to support Oracle GoldenGate capture and apply must be enabled explicitly for all Oracle database versions. This is required for all Extract and Replicat modes.
- [Setting Flashback Query](#)
To process certain update records, Extract fetches additional row data from the source database.
- [Managing Server Resources](#)
In integrated mode, Extract interacts with an underlying logmining server in the source database and Replicat interacts with an inbound server in the target database. This section provides guidelines for managing the shared memory consumed by these servers.
- [Ensuring Row Uniqueness in Source and Target Tables](#)
Oracle GoldenGate requires a unique row identifier on the source and target tables to locate the correct target rows for replicated updates and deletes.

Configuring Connections for Integrated Processes

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

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

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

```
TEST =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP) (HOST = test2) (PORT = 1521))
```

```

    )
(CONNECT_DATA =
  (SERVER = DEDICATED)
  (SERVICE_NAME = test)
)
)

```

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

Password encryption method:

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

Credential store method:

```
USERIDALIAS ext
```

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

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

Configuring Logging Properties

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

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

Note:

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

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

Logging option	GGSCI command	What it does	Use case
Forced logging mode	ALTER DATABASE FORCE LOGGING;	Forces the logging of all transactions and loads.	Strongly recommended for all Oracle GoldenGate use cases. <code>FORCE LOGGING</code> overrides any table-level <code>NOLOGGING</code> settings.

Logging option	GGSCI command	What it does	Use case
Minimum database-level supplemental logging	ALTER DATABASE ADD SUPPLEMENTAL LOG DATA	Enables minimal supplemental logging to add row-chaining information to the redo log.	Required for all Oracle GoldenGate use cases
Schema-level supplemental logging, default setting See Enabling Schema-level Supplemental Logging .	ADD SCHEMATRANDATA	Enables unconditional supplemental logging of the primary key and conditional supplemental logging of unique key(s) and foreign key(s) of all tables in a schema. All of these keys together are known as the <i>scheduling columns</i> .	Enables the logging for all current and future tables in the schema. If the primary key, unique key, and foreign key columns are not identical at both source and target, use ALLCOLS. Required when using <i>scheduling columns</i> . DDL support.
Schema-level supplemental logging with unconditional logging for all supported columns. (See Enabling Schema-level Supplemental Logging for non-supported column types.)	ADD SCHEMATRANDATA with 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 Enabling Table-level Supplemental Logging	ADD TRANDATA	Enables unconditional supplemental logging of the primary key and conditional supplemental logging of unique key(s) and foreign key(s) of a table. All of these keys together are known as the <i>scheduling columns</i> .	Required for all Oracle GoldenGate use cases unless schema-level supplemental logging is used. If the primary key, unique key, and foreign key columns are not identical at both source and target, use ALLCOLS.

Logging option	GGSCI command	What it does	Use case
Table-level supplemental logging with unconditional logging for all supported columns. (See Enabling Table-level Supplemental Logging for non-supported column types.)	ADD TRANDATA with ALLCOLS option	Enables unconditional supplemental logging of all of the columns of the table.	Used for bidirectional and active-active configurations where all column values are checked, not just the changed columns, when attempting to perform an update or delete. This takes more resources though allows for the highest level of real-time data validation and thus conflict detection. It can also be used when the source and target primary, unique, and foreign keys are not the same or are constantly changing between source and target.
Table-level supplemental logging, minimal setting	ADD TRANDATA with NOSCHEDULINGCOLS option	Enables unconditional supplemental logging of the primary key and all valid unique indexes of a table.	Use for nonintegrated Replicat and non-parallel Replicat. This is the minimum required table-level logging.



Note:

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

- [Enabling Minimum Database-level Supplemental Logging](#)
- [Enabling Schema-level Supplemental Logging](#)
- [Enabling Table-level Supplemental Logging](#)

Enabling Minimum Database-level Supplemental Logging

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

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

 **Note:**

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

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

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

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

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

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

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

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

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

5. Switch the log files.

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

Enabling Schema-level Supplemental Logging

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

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

 **Note:**

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

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

1. Run GGSCI 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 *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.

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

Where:

- Without options, `ADD SCHEMATRANDATA` schema enables the unconditional supplemental logging on the source system of the primary key and the conditional supplemental logging of all unique key(s) and foreign key(s) of all current and future tables in the given schema. Unconditional logging forces the primary key values to the log whether or not the key was changed in the current operation. Conditional logging logs all of the column values of a foreign or unique key if at least one of them was changed in the current operation. The default is optional to support nonintegrated Replicat but is required to support integrated Replicat because primary key, unique keys, and foreign keys must all be available to the inbound server to compute dependencies. For more information about integrated Replicat, see [Deciding Which Apply Method to Use](#).
- `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 `finance` schema.

```
ADD SCHEMATRANDATA MY_PDB.FINANCE 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 MY_PDB.HR NOSCHEDULINGCOLS
```

Enabling Table-level Supplemental Logging

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

- To enable the required level of logging when not using schema-level logging (see [Enabling 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 GGSCI 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 *Reference for Oracle GoldenGate* for more information about `DBLOGIN` and additional options.

3. Issue the `ADD TRANDATA` command.

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

Where:

- *PDB* is the name of the root container or pluggable database if the table is in a multitenant container database.
- *schema* is the source schema that contains the table.
- *table* is the name of the table. See *Specifying Object Names in Oracle GoldenGate Input* in *Administering Oracle GoldenGate* for instructions for specifying object names.
- `ADD TRANDATA` without other options automatically enables unconditional supplemental logging of the primary key and conditional supplemental logging of unique key(s) and foreign key(s) of the table. Unconditional logging forces the primary key values to the log whether or not the key was changed in the current operation. Conditional logging logs all of the column values of a foreign or unique key if at least one of them was changed in the current operation. The default is optional to support nonintegrated Replicat (see also `NOSCHEDULINGCOLS`) but is required to support integrated Replicat because primary key, unique keys, and foreign keys must all be available to the inbound server to compute dependencies. For more information about integrated Replicat, see [Deciding Which Apply Method to Use](#).
- `ALLCOLS` enables the unconditional supplemental logging of all of the columns of the table. Use to support integrated Replicat when the source and target tables have

different scheduling columns. (*Scheduling columns* are the primary key, the unique key, and the foreign key.)

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

Enabling Oracle GoldenGate in the Database

The database services required to support Oracle GoldenGate capture and apply must be enabled explicitly for all Oracle database versions. This is required for all Extract and 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 process certain update records, Extract fetches additional row data from the source database.

Oracle GoldenGate fetches data for the following:

- User-defined types
- Nested tables
- XMLType objects

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

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

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

```
UNDO_MANAGEMENT=AUTO
```

```
UNDO_RETENTION=86400
```

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

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

$$undo_space = UNDO_RETENTION * UPS + overhead$$

Where:

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

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

3. For tables that contain LOBs, do one of the following:
 - Set the `LOB` storage clause to `RETENTION`. This is the default for tables that are created when `UNDO_MANAGEMENT` is set to `AUTO`.
 - If using `PCTVERSION` instead of `RETENTION`, set `PCTVERSION` to an initial value of 25. You can adjust it based on the fetch statistics that are reported with the `STATS EXTRACT` command. If the value of the `STAT_OPER_ROWFETCH_CURRENTBYROWID` or `STAT_OPER_ROWFETCH_CURRENTBYKEY` field in these statistics is high, increase `PCTVERSION` in increments of 10 until the statistics show low values.
4. Grant either of the following privileges to the Oracle GoldenGate Extract user:

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

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

Oracle GoldenGate provides the following parameters to manage fetching.

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

Parameter or Command	Description
REFFETCHEDCOLOPTIONS parameter	Controls the response by Replicat when it processes trail records that include fetched data or column-missing conditions.

Managing Server Resources

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

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

By default, one integrated capture Extract requests the logmining server to run with `MAX_SGA_SIZE` of 1GB. Thus, if you are running three Extracts in integrated capture mode 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 in integrated capture mode, set `STREAMS_POOL_SIZE` for the database to the following value:

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

Ensuring Row Uniqueness in Source and Target Tables

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

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

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

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

- First unique key alphanumerically with no virtual columns, no UDTs, no function-based columns, no nullable columns, and no extended (32K) `VARCHAR2/NVARCHAR2` columns. To support a key that contains columns that are part of an invisible index, you must use the `ALLOWINVISIBLEINDEXKEYS` parameter in the Oracle GoldenGate `GLOBALS` file.
- First unique key alphanumerically with no virtual columns, no UDTs, no extended (32K) `VARCHAR2/NVARCHAR2` columns, or no function-based columns, but can include nullable columns. To support a key that contains columns that

are part of an invisible index, you must use the `ALLOWINVISIBLEINDEXKEYS` parameter in the Oracle GoldenGate `GLOBALS` file.

3. Not Nullable Unique keys: At least one column from one of the unique keys must be not nullable. This is because `NOALLOWNULLABLEKEYS` is the default.

 **Note:**

`ALLOWNULLABLEKEYS` is not valid for integrated Replicat.

4. 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 *Reference for Oracle GoldenGate*.

3

Establishing Oracle GoldenGate Credentials

Learn how to create database users for the processes that interacts with the database, assign the correct privileges, and secure the credentials from unauthorized use.

Topics

- [Assigning Credentials to Oracle GoldenGate](#)
The Oracle GoldenGate processes require one or more database credentials with the correct database privileges for the database version, database configuration, and Oracle GoldenGate features that you are using.
- [Securing the Oracle GoldenGate Credentials](#)
To preserve the security of your data, and to monitor Oracle GoldenGate processing accurately, do not permit other users, applications, or processes to log on as, or operate as, an Oracle GoldenGate database user.

Assigning Credentials to Oracle GoldenGate

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

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

See [Creating and Populating the Credential Store](#) to learn about creating and using the credential store.

- [Oracle GoldenGate Users \(Database\)](#)

Oracle GoldenGate Users (Database)

A user is required in the source database for the Service Manager in MA or the Manager process in CA if you are using Oracle GoldenGate DDL support. This user performs maintenance on the Oracle GoldenGate database objects that support DDL capture.

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

Additional users or privileges may be required to use the following features, if Extract will run in classic capture mode:

- RMAN log retention, see [Log Retention Options](#).
- TDE support, see [Configuring Oracle TDE Data in Classic Capture Mode](#).
- ASM, see [Mining ASM-stored Logs in Classic Capture Mode](#).

- [Granting the Appropriate User Privileges](#)

Granting the Appropriate User Privileges

The user privileges that are required for Oracle GoldenGate depend on the database version and the Extract or Replicat process mode. For more information about process modes, see [Choosing Capture and Apply Modes](#).

- [Oracle Database Privileges](#)
- [About the `dbms_goldengate_auth.grant_admin_privilege` Package](#)
- [Optional Grants for `dbms_goldengate_auth.grant_admin_privilege`](#)

Oracle Database Privileges

The following privileges apply to Oracle database.

Privilege	Extract	Replicat All Modes	Purpose
CREATE SESSION	No	No	Connect to the database
CREATE VIEW	No	No	Required to add the heartbeat table view.
RESOURCE	No	No	Create objects. In Oracle Database 12cR1 and later, instead of RESOURCE, grant the following privilege: <pre>ALTER USER user QUOTA {size UNLIMITED} ON tablespace;</pre>
ALTER SYSTEM	No	No	Perform administrative changes, such as enabling logging.
ALTER USER	No	No	Required for multitenant architecture and <i>GGADMIN</i> should be a valid Oracle GoldenGate administrator schema.
EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE ('REPUSER', CONTAINER=>'PDB1');	Yes	Yes	This is 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.
Privileges granted through DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE	No	No	(Extract) Grants privileges for Extract, including the logging server. (Replicat) Grants privileges for both non-integrated and integrated Replicat, including the database inbound server.

Privilege	Extract	Replicat All Modes	Purpose
Any or all of optional privileges of DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE	No	No	<ul style="list-style-type: none"> Capture from Virtual Private Database Capture redacted data See About the dbms_goldengate_auth.grant_admin_privilege Package for more information.
Grant the following privileges connected as SYS user to Extract and Replicat users: EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE('ggadmin user', '*', GRANT_OPTIONAL_PRIVILEGES=>'*'); GRANT DV_GOLDENGATE_ADMIN, DV_GOLDENGATE_REDO_ACCESS to 'ggadmin user';	No	No	Capture from Data Vault
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 scipts, BEGIN DVSYS.DBMS_MACADM.ADD_AUTH_TO_REALM(REALM_NAME => 'Oracle Default Component Protection Realm', GRANTEE => 'GGADMIN USER', AUTH_OPTIONS => 1) ; END ; / EXECUTE DBMS_MACADM.AUTHORIZE_DDL('SYS', 'SYSTEM');	No	No	Capture from Data Vault
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
INSERT, UPDATE, DELETE on target tables	NA	No	Apply replicated DML to target objects

Privilege	Extract	Replicat All Modes	Purpose
GRANT INSERT ANY TO..., GRANT UPDATE ANY TO... and GRANT DELETE ANY TO...	NA	No	Use these privileges for the Replicat user, instead of granting INSERT, UPDATE, DELETE to every table, if replicating every table.
DDL privileges on target objects (if using DDL support)	NA	No	Issue replicated DDL on target objects
LOCK ANY TABLE	NA	No	Lock target tables. Only required for initial load using direct bulk load to SQL*Loader.
SELECT ANY DICTIONARY	No	No	Allow all privileges to work properly on dictionary tables.
SELECT ANY TRANSACTION	No	NA	Use a newer Oracle ASM API.

Here's an example of the list of permissions granted for the Oracle database root container:

```
DROP USER c##ggadmin CASCADE;
CREATE USER c##ggadmin IDENTIFIED BY passw0rd CONTAINER=all DEFAULT
TABLESPACE GG_DATA TEMPORARY TABLESPACE temp;
ALTER USER c##ggadmin SET CONTAINER_DATA=all CONTAINER=current;
GRANT CREATE SESSION to c##ggadmin;
GRANT CREATE VIEW to c##ggadmin;
GRANT CONNECT to c##ggadmin CONTAINER=all;
GRANT RESOURCE to c##ggadmin;
GRANT ALTER SYSTEM to c##ggadmin ;
GRANT SELECT ANY DICTIONARY to c##ggadmin ;
EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE('c##ggadmin');
ALTER USER c##ggadmin QUOTA unlimited ON GG_DATA;

SELECT * FROM DBA_SYS_PRIVS WHERE GRANTEE='c##ggadmin' ORDER BY 2;
```

In this example, DBA privilege is not provided but the user will be able to access the DBA_SYS_PRIVS package.

About the dbms_goldengate_auth.grant_admin_privilege Package

Most of the privileges that are needed for Extract and Replicat to operate in classic and integrated mode are granted through the dbms_goldengate_auth.grant_admin_privilege package.

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

```
grant_admin_privilege('ggadm')
grant_admin_privilege('ggadm','capture');
```

The following example shows Extract on Oracle 12c Multitenant Database:

```
BEGIN
dbms_goldengate_auth.grant_admin_privilege
( grantee => 'C##GGADMIN', privilege_type => 'CAPTURE',
  grant_select_privileges => TRUE, do_grants => TRUE, container => 'ALL'
);
END;
```

Optional Grants for dbms_goldengate_auth.grant_admin_privilege

This procedure grants the privileges needed by a user to be a Oracle GoldenGate administrator.

See `DBMS_GOLDENGATE_AUTH` in *Oracle Database PL/SQL Packages and Types Reference*

Securing the Oracle GoldenGate Credentials

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

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

The credential store stores the user name and password for each of the assigned Oracle GoldenGate users. A user ID is associated with one or more aliases, and it is the alias that is supplied in commands and parameter files, not the actual user name or password. The credential file can be partitioned into domains, allowing a standard set of aliases to be used for the processes, while allowing the administrator on each system to manage credentials locally.

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

4

Choosing Capture and Apply Modes

This chapter contains information that helps you determine the appropriate capture and apply modes for your database environment.

Topics:

- [Overview of Oracle GoldenGate Capture and Apply Processes](#)
The Oracle GoldenGate capture process is known as *Extract*. Each instance of an Extract process is known as a *group*, which includes the process itself and the associated files that support it.
- [Deciding Which Capture Method to Use](#)
For an Oracle source database, you can run Extract in either *integrated capture* or *classic capture* mode.
- [Deciding Which Apply Method to Use](#)
The Replicat process is responsible for the application of replicated data to an Oracle target database.
- [Using Different Capture and Apply Modes Together](#)
You can use integrated capture and classic capture concurrently within the same source Oracle GoldenGate instance, and you can use integrated Replicat and nonintegrated Replicat concurrently within the same target Oracle GoldenGate instance. This configuration requires careful placement of your objects within the appropriate process group, because there is no coordination of DDL or DML between classic and integrated capture modes, nor between nonintegrated and integrated Replicat modes.
- [Switching to a Different Process Mode](#)
You can switch between process modes. For example, you can switch from classic capture to integrated capture, or from integrated capture to classic capture.

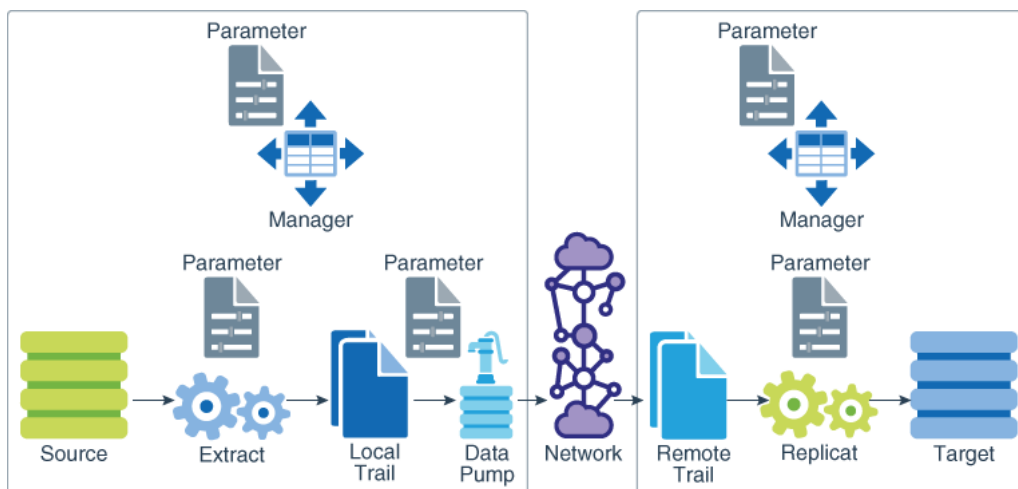
Overview of Oracle GoldenGate Capture and Apply Processes

The Oracle GoldenGate capture process is known as *Extract*. Each instance of an Extract process is known as a *group*, which includes the process itself and the associated files that support it.

An additional Extract process, known as a *data pump*, is recommended to be used on the source system, so that captured data can be persisted locally to a series of files known as a *trail*. The data pump does not capture data but rather reads the local trail and propagates the data across the network to the target.

The Oracle GoldenGate apply process is known as *Replicat*. Each instance of a Replicat process is known as a *group*, which includes the process itself and the associated files that support it. Replicat reads data that is sent to local storage, known as a *trail*, and applies it to the target database.

The following diagram illustrates the basic Oracle GoldenGate process configuration.



Note:

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

Deciding Which Capture Method to Use

For an Oracle source database, you can run Extract in either *integrated capture* or *classic capture* mode.

Although you can use the classic capture mode, it is recommended that you use the integrated capture mode because classic capture has been deprecated and is not being enhanced for any future releases. It will be desupported in future releases and any classic Extract configuration will need to be migrated to integrated Extract.

The method that you use determines how you configure the Oracle GoldenGate processes and depends on such factors as:

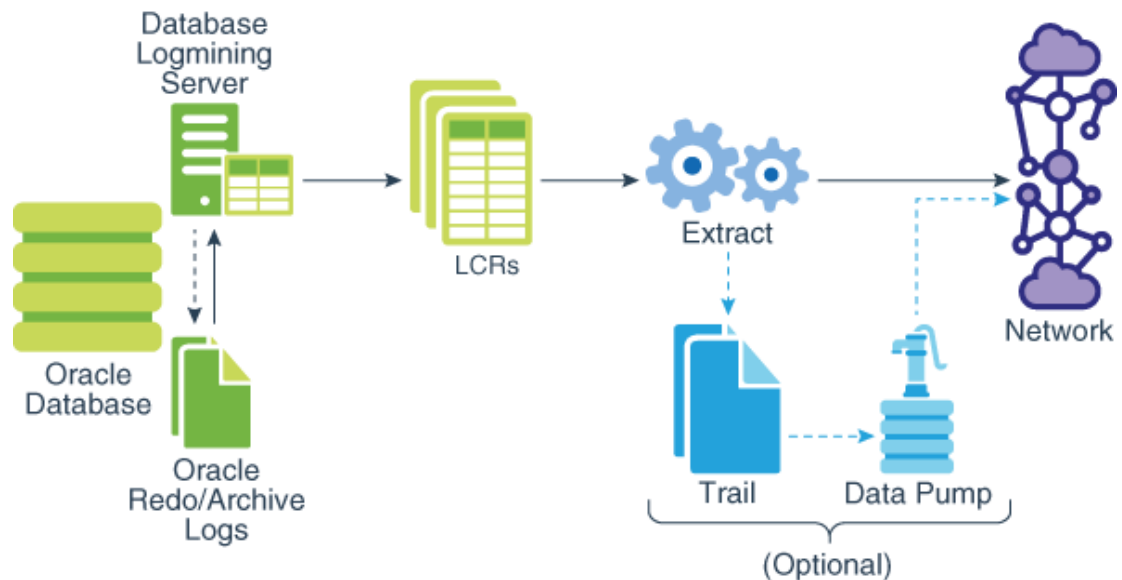
- the data types involved
- the database configuration
- the version of the Oracle Database

The following explains these modes and the database versions that each mode supports.

- [About Integrated Capture](#)
- [About Classic Capture](#)

About Integrated Capture

In integrated capture mode, the Oracle GoldenGate Extract process interacts directly with a database logmining server to receive data changes in the form of logical change records (LCR). The following diagram illustrates the configuration of Extract in integrated capture mode.



Integrated capture supports more data and storage types as compared to classic capture, and the support is more transparent. For more information, see [Summary of Supported Oracle Data Type and Objects Per Capture Mode](#).

The following are some additional benefits of integrated capture:

- Because integrated capture is fully integrated with the database, no additional setup is required to work with Oracle RAC, ASM, and TDE.
- Integrated capture uses the database logmining server to access the Oracle redo stream, with the benefit of being able to automatically switch between different copies of archive logs or different mirrored versions of the online logs. Thus integrated capture can transparently handle the absence of a log file caused by disk corruption, hardware failure, or operator error, assuming that additional copies of the archived and online logs are available
- Integrated capture enables faster filtering of tables.
- Integrated capture handles point-in-time recovery and RAC integration more efficiently.
- Integrated capture features integrated log management. The Oracle Recovery Manager (RMAN) automatically retains the archive logs that are needed by Extract.
- Integrated capture is the only mode that supports capture from a multitenant container database. One Extract can mine multiple pluggable databases within a multitenant container database.
- For a release 11.2.0.4 source database and later (with source compatibility set to 11.2.0.4 or higher), the capture of DDL is performed by the logmining server asynchronously and requires no special triggers, tables, or other database objects to be installed. Oracle GoldenGate upgrades can be performed without stopping user applications. The use of a DDL trigger and supporting objects is required when Extract is in integrated mode with an Oracle 11g source database that is earlier than version 11.2.0.4.
- Because integrated capture and integrated apply are both database objects, the naming of the objects follow the same rules as other Oracle Database objects, see [Specifying Object Names in Oracle GoldenGate Input in Administering Oracle GoldenGate](#).
- [Integrated Capture Deployment Options](#)

Integrated Capture Deployment Options

The deployment options for integrated capture are described in this section and depend on where the mining database is deployed. The mining database is the one where the logmining server is deployed.

- **Local deployment:** For a local deployment, the source database and the mining database are the same. The source database is the database for which you want to mine the redo stream to capture changes, and also where you deploy the logmining server. Because integrated capture is fully integrated with the database, this mode does not require any special database setup.
- **Downstream deployment:** In a downstream deployment, the source and mining databases are different databases. You create the logmining server at the downstream database. You configure redo transport at the source database to ship the redo logs to the downstream mining database for capture at that location. Using a downstream mining server for capture may be desirable to offload the capture overhead and any other overhead from transformation or other processing from the production server, but requires log shipping and other configuration.

When using a downstream mining configuration, the source database and mining database must be of the same platform. For example, if the source database is running on Windows 64-bit, the downstream database must also be on a Windows 64-bit platform. See [Configuring a Downstream Mining Database](#) and [Example Downstream Mining Configuration](#) to configure a downstream mining database.

- **Downstream sourceless Extract deployment:** In the Extract parameter file, replace the `USERID` parameter with `NOUSERID`. You must use `TRANLOGOPTIONS MININGUSER`. Extract obtains all required information from the downstream mining database. Extract is not dependent on any connection to the source database. The source database can be shutdown and restarted without affecting Extract.

Extract will abend if it encounters redo changes that require data to be fetched from the source database.

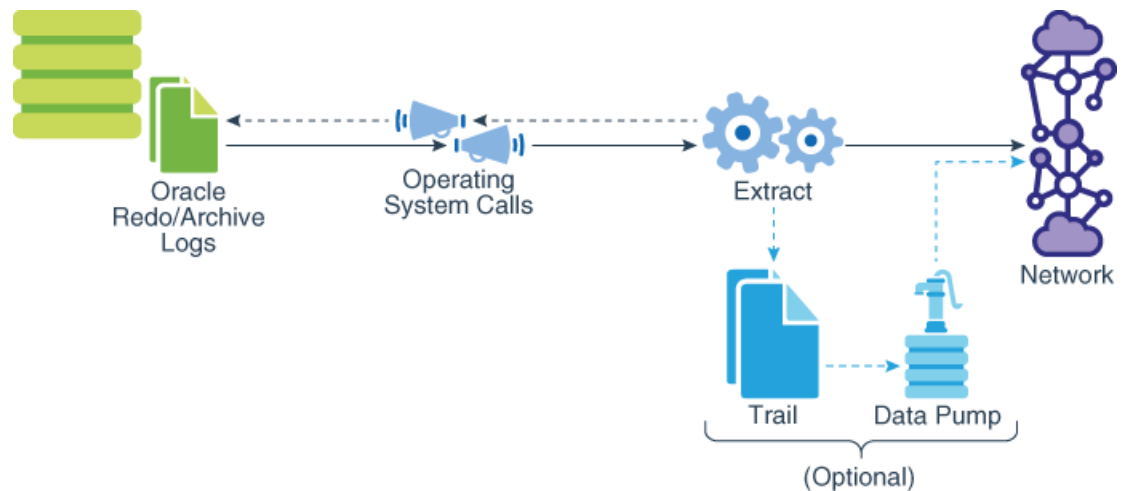
To capture any tables that are listed as `ID KEY` in the `dba_goldengate_support_mode` view, you need to have a `FETCHUSERID` or `FETCHUSERIDALIAS` connection to the support the tables. Tables that are listed as `FULL` do not require this. We also need to state that if a customer wants to perform `SQLEXEC` operations that perform a query or execute a stored procedure they cannot use this method as it is incompatible with `NOUSERID` because `SQLEXEC` works with `USERID` or `USERIDALIAS`.

About Classic Capture

In classic capture mode, the Oracle GoldenGate Extract process captures data changes from the Oracle redo or archive log files on the source system or from shipped archive logs on a standby system. The following diagram illustrates the configuration of an Extract in classic capture mode.

Note:

Classic capture has been deprecated from Oracle GoldenGate 18c (18.1.0) and higher releases.



Classic capture supports most Oracle data types fully, with restricted support for the complex data types. Classic capture is the original Oracle GoldenGate capture method. You can use classic capture for any source Oracle RDBMS that is supported by Oracle GoldenGate, with the exception of the multitenant container database.

For more information, see [Details of Support for Oracle Data Types and Objects](#).

Deciding Which Apply Method to Use

The Replicat process is responsible for the application of replicated data to an Oracle target database.

For an Oracle target database, you can run Replicat in parallel, non-integrated or integrated mode. Oracle recommends that you use the parallel Replicat unless a specific feature requires a different type of Replicat.

Topics:

- [About Parallel Replicat](#)
- [About Non-integrated Replicat](#)
- [About the Integrated Replicat Mode](#)

About Parallel Replicat

Parallel Replicat is a new 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.

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
Auto-parallelism	Yes	Yes	No	No
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

Parallel Replicat supports all databases using the non-integrated option. Parallel Replicat only supports replicating data from trails with full metadata, which requires the classic trail format.

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 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.

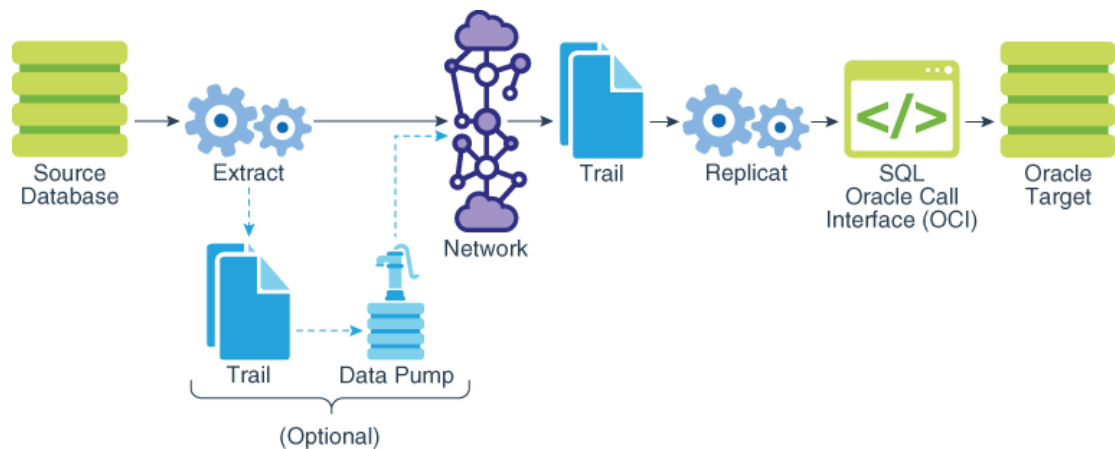
- Appliers reorder records within a batch for array execution. It applies the batch to the target database and performs error handling. It also tracks applied transactions in checkpoint tables.

About Non-integrated Replicat

In non-integrated mode, the Replicat process uses standard SQL to apply data directly 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 Oracle Call Interface (OCI).

The following diagram illustrates the configuration of Replicat in non-integrated mode.



Use non-integrated Replicat when you want to make heavy use of features that are not supported in integrated Replicat mode, see [About the Integrated Replicat Mode](#).

You can apply transactions in parallel with a non-integrated Replicat by using a coordinated Replicat configuration.

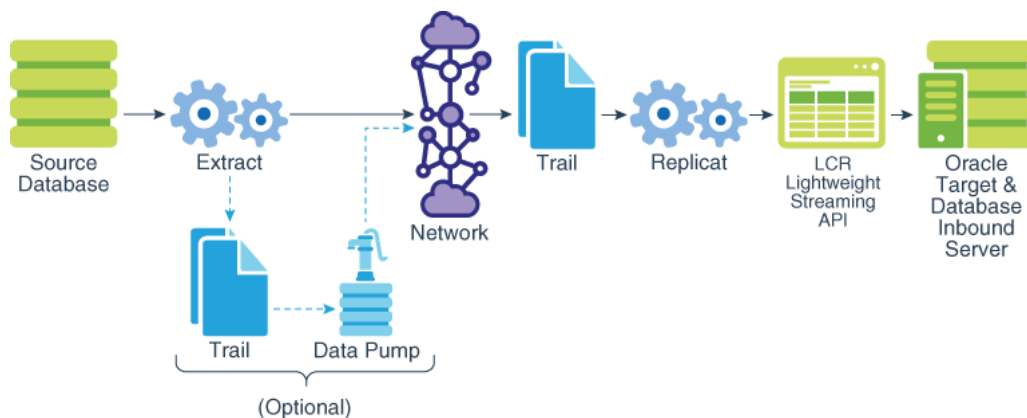
About the Integrated Replicat Mode

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:

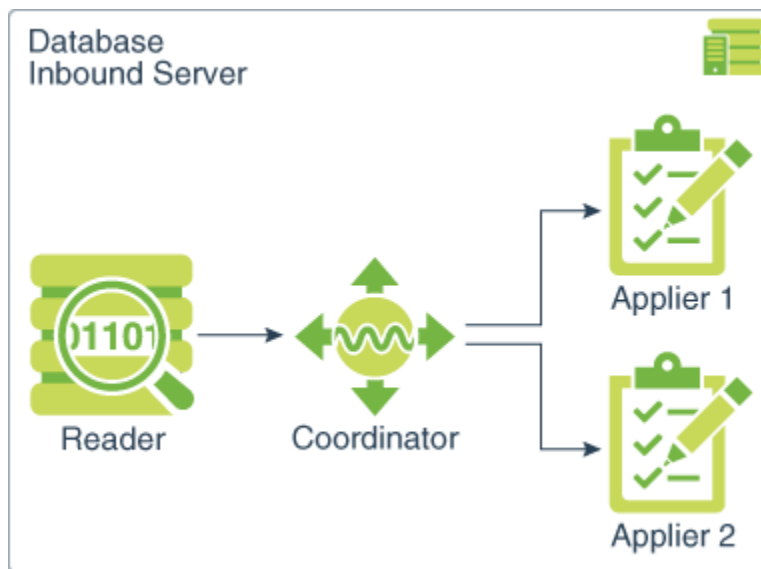
- Reads the Oracle GoldenGate trail.
- Performs data filtering, mapping, and conversion.
- Constructs logical change records (LCR) that represent source database DML transactions (in committed order). DDL is applied directly by Replicat.
- Attaches to a background process in the target database known 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.

The following figure illustrates the configuration of Replicat 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. 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.



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
- `SOLEXEC` parameter within a `TABLE` or `MAP` parameter
- `EVENTACTIONS` processing
- UDT Note, if the extract uses `USENATIVEOBSUPPORT` to capture the UDT, then integrated Replicat will apply it with the inbound server, otherwise it will be handled by Replicat directly.

Because transactions are applied serially in direct apply mode, 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, see *Monitoring and Controlling Processing After the Instantiation in Using Oracle GoldenGate for Oracle Database*.

 **Note:**

User exits are executed in integrated mode. The user exit may produce unexpected results, however, if the exit code depends on data in the replication stream.

- [Benefits of Integrated Replicat](#)
- [Integrated Replicat Requirements](#)

Benefits of Integrated Replicat

The following are the benefits of using integrated Replicat versus nonintegrated 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 works with single or pluggable databases.

Integrated Replicat Requirements

To use integrated Replicat, the following must be true.

- The target Oracle Database must be Oracle 11.2.0.4 or later.
- 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.

Using Different Capture and Apply Modes Together

You can use integrated capture and classic capture concurrently within the same source Oracle GoldenGate instance, and you can use integrated Replicat and nonintegrated Replicat concurrently within the same target Oracle GoldenGate instance. This configuration requires careful placement of your objects within the appropriate process group, because there is no coordination of DDL or DML between classic and integrated capture modes, nor between nonintegrated and integrated Replicat modes.

Each Extract group must process objects that are suited to the processing mode, based on table data types and attributes. No objects in one Extract can have DML or DDL dependencies on objects in the other Extract. The same type of segregation must be applied to the Replicat configuration.

You can use the following capture and apply modes together:

- Classic capture (Oracle or non-Oracle source) and nonintegrated Replicat
- Classic capture (Oracle or non-Oracle source) and integrated Replicat
- Integrated capture and nonintegrated Replicat
- Integrated capture and integrated Replicat

The recommended Oracle GoldenGate configuration, when supported by the Oracle version, is to use one integrated capture on an Oracle source and one integrated Replicat per source database on an Oracle target. Integrated capture supports certain data types more completely than classic capture. One integrated Replicat configuration supports all Oracle data types either through the inbound server or by switching to direct apply when necessary, and it preserves source transaction integrity. You can adjust the parallelism settings to the desired apply performance level as needed.

If the target database is an Oracle version that does not support integrated Replicat, or if it is a non-Oracle Database, you can use a coordinated Replicat configuration. See *Administering Oracle GoldenGate* for more information.

Switching to a Different Process Mode

You can switch between process modes. For example, you can switch from classic capture to integrated capture, or from integrated capture to classic capture.

For instructions, see Performing Administrative Operations in *Administering Oracle GoldenGate*.

5

Using Parallel Replicat

You can create (or add) and configure parallel replication in your environment. New Parallel Replicat processes then process the information in all the internal stages, from the beginning to the end in parallel. Components, such as Mappers, Master, and Appliers are also explained.

To know more about parallel Replicat and the parallel replication architecture, see About Parallel Replicat.

Topics:

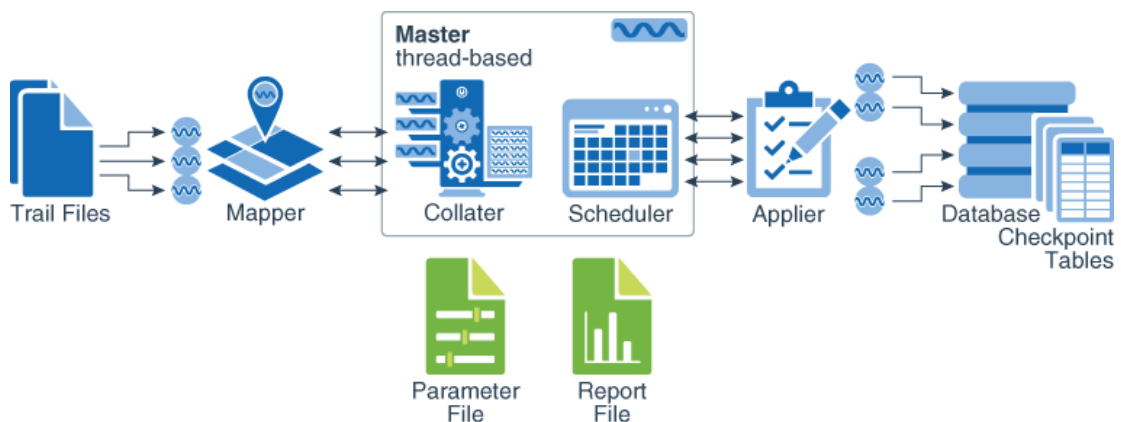
- [Parallel Replication Architecture](#)
Parallel replication processes leverage the apply processing functionality that is available within the Oracle Database in integrated mode.
- [Basic Parameters for Parallel Replicat](#)
The following table lists the basic parallel Replicat parameters and their description.
- [Creating a Parallel Replicat](#)
You can create a parallel replication using the graphical user interface or the command line interfaces GGSCI and the Admin Client.

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 architecture diagram depicts the flow of change records through the various processes of a parallel replication from the trail files to the target database.



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 <i>not</i> 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 repA
userid ggadmin, password ***
MAP_PARALLELISM 3
MIN_APPLY_PARALLELISM 2
MAX_APPLY_PARALLELISM 10
SPLIT_TRANS_RECS 1000
map *.* , target *.*;
```

Creating a Parallel Replicat

You can create a parallel replication using the graphical user interface or the command line interfaces GGSCI and the Admin Client.

A parallel Replicat requires a checkpoint table so both the Administration Server UI and Admin Client issue an error when the parallel Replicat does not include a checkpoint table.



Note:

Parallel replication does not support `COMMIT_SERIALIZATION` in Integrated Mode. To use this apply process, use Integrated Replicat.

Creating a Non-Integrated Parallel Replication with the Administration Server

1. Open a browser and connect to the Service Manager that you created with the Configuration Assistant:

```
https://server_name:service_manger_port/
```

For Example, `https://localhost:9000/`. In a non-secured environment, use `http` instead of `https`.

The Oracle GoldenGate Service Manager is displayed.

2. Enter the username and password you created and click **Sign In**.
In the Service Manager, you can see servers that are running.
3. In the Services section, click **Administration Server**, and then log in.
4. Click the Application Navigation icon to the left of the page title to expand the navigation panel.
5. Create the checkpoint table by clicking **Configuration** in the right navigation panel.
6. Ensure that you have a valid credential and log in to the database by clicking the 'log in database' icon under **Action**.
7. Click the **+** sign to add a checkpoint table.
8. Enter the `schema.name` of the checkpoint table that you would like to create, and then click **Submit**.
9. Validate that the table was created correctly by logging out of the Credential Alias using the log out database icon, and then log back in.

Once the log in is complete, your new checkpoint table is listed.

10. Click **Overview** to return to the main Administration Server page.
11. Click the **+** sign next to **Replicats**.
12. Select **Nonintegrated Replicat** then click **Next**.
13. Enter the required information making sure that you complete the Credential Domain and Credential Alias fields before completing the Checkpoint Table field, and then select your newly created Checkpoint Table from the list.
14. Click **Next**, and then click **Create and Run** to complete the Replicat creation.

Creating a Non-Integrated Parallel Replicat with the Admin Client

1. Go the `bin` directory of your Oracle GoldenGate installation directory.

```
cd $OGG_HOME/bin
```

2. Start the Admin Client.

```
./adminclient
```

The Admin Client command prompt is displayed.

```
OGG (not connected) 12>
```

3. Connect to the Service Manager deployment source:

```
connect http://localhost:9500 deployment Target1 as oggadmin password  
welcome1
```

You must use `http` or `https` in the connection string; this example is a non-SSL connection.

4. Add the Parallel Replicat, which may take a few minutes to complete:

```
add replicat R1, parallel, exttrail bb checkpointtable ggadmin.ggcheckpoint
```

You could use just the two character trail name as part of the `ADD REPLICAT` or you can use the full path, such as `/u01/oggdeployments/target1/var/lib/data/bb`.

5. Verify that the Replicat is running:

```
info replicat R1
```

Messages similar to the following are displayed:

```
REPLICAT   R1           Initialized   2016-12-20 13:56   Status RUNNING  
Parallel  
Checkpoint Lag      00:00:00 (updated 00:00:22 ago)  
Process ID          30007  
Log Read  
Checkpoint File ./ra000000000First Record RBA 0
```

6

Configuring Capture in Integrated Mode

This chapter contains instructions for configuring the Oracle GoldenGate capture process to capture transaction data in integrated mode.



Note:

To switch an active Extract configuration from classic to integrated mode, perform these configuration steps and then see *Administering Oracle GoldenGate*.

In case the Integrated Extract is running from a remote system, Oracle GoldenGate automatically enables cross endian interoperability. This implies that if the endian value where Integrated Extract is running is different from the endian value where the Oracle database is running, then the cross endian support is automatically enabled.

Topics:

- [Prerequisites for Configuring Integrated Capture](#)
You must adhere to the guidelines provided in this topic before configuring an Extract in integrated mode.
- [What to Expect from these Instructions](#)
These instructions show you how to configure a basic Extract parameter (configuration) file for the primary Extract, which captures transaction data from the data source, and for a data-pump Extract, which propagates captured data that is stored locally in a *trail* from the source system to the target system.
- [Configuring the Primary Extract in Integrated Capture Mode](#)
The mining database from which the primary Extract captures log change records from the logmining server, can be either local or downstream from the source database.
- [Configuring the Data Pump Extract](#)
A data pump can perform data filtering, mapping, and conversion, or it can be configured in pass-through mode, where data is passively transferred as-is, without manipulation.
- [Next Steps](#)
A parameter file is a plain text file that is read by an associated Oracle GoldenGate process. Oracle GoldenGate uses two types of parameter files: a GLOBALS file and runtime parameter files.

Prerequisites for Configuring Integrated Capture

You must adhere to the guidelines provided in this topic before configuring an Extract in integrated mode.

The guidelines for configuring an Extract in integrated mode are:

1. [Preparing the Database for Oracle GoldenGate.](#)
2. [Establishing Oracle GoldenGate Credentials.](#)

3. [Choosing Capture and Apply Modes](#).
4. Create the Oracle GoldenGate instance on the source system by configuring the Manager process. See *Administering Oracle GoldenGate*.
5. Additionally, review the guidelines in *Administering Oracle GoldenGate*.

What to Expect from these Instructions

These instructions show you how to configure a basic Extract parameter (configuration) file for the primary Extract, which captures transaction data from the data source, and for a data-pump Extract, which propagates captured data that is stored locally in a *trail* from the source system to the target system.

Your business requirements probably will require a more complex topology, but this procedure forms a basis for the rest of your configuration steps.

By performing these steps, you can:

- get the basic configuration files established.
- build upon them later by adding more parameters as you make decisions about features or requirements that apply to your environment.
- use copies of them to make the creation of additional parameter files faster than starting from scratch.

Configuring the Primary Extract in Integrated Capture Mode

The mining database from which the primary Extract captures log change records from the logmining server, can be either local or downstream from the source database.

These steps configure the primary Extract to capture transaction data in integrated mode from either location. See [Configuring a Downstream Mining Database](#) and [Example Downstream Mining Configuration](#) for more information about capturing from a downstream mining database.

Note:

One Extract group is generally sufficient to capture from a single database or multiple pluggable databases within a multitenant container database. See [Configuring Oracle GoldenGate in a Multitenant Container Database](#).

1. In GGSCI on the source system, create the Extract parameter file.

```
EDIT PARAMS name
```

Where: *name* is the name of the primary Extract.

Note:

To learn about using Oracle GoldenGate microservices to perform this task, see [How to Add Extracts](#).

- Enter the Extract parameters in the order shown, starting a new line for each parameter statement. Examples are shown for a regular database, a multitenant container database, and downstream deployments for both non-CDB and multitenant databases. The difference between the two is whether you must use two-part or three-part object names in the TABLE and SEQUENCE specifications. See the Basic Parameters for primary Extract (classic or integrated mode) for more information and parameter descriptions.

Basic parameters for Extract mining a non-multitenant database

```
EXTRACT financep
USERIDALIAS c##_alias
DDL INCLUDE MAPPED
EXTTRAIL /ggs/dirdat/lt
SEQUENCE hr.employees_seq;
TABLE hr.*;
```

Basic parameters for Integrated Extract capturing from a multitenant database

```
EXTRACT financep
USERIDALIAS c##_alias
DDL INCLUDE MAPPED
EXTTRAIL /ggs/dirdat/lt
TABLE test.ogg.tab1;
SEQUENCE hr.employees_seq;
TABLE hr.*;
TABLE sales.*;
TABLE acct.*;
```

Basic parameters for Extract where the mining database is a downstream database and is a non-CDB database

```
EXTRACT financep
USERIDALIAS c##_alias
TRANLOGOPTIONS MININGUSERALIAS c##_alias
TRANLOGOPTIONS INTEGRATEDPARAMS (DOWNSTREAM_REAL_TIME_MINE Y)
LOGALLSUPCOLS
UPDATERECORDFORMAT COMPACT
DDL INCLUDE MAPPED
ENCRYPTTRAIL AES192
EXTTRAIL /ggs/dirdat/lt
SEQUENCE hr.employees_seq;
TABLE hr.*;
```

Basic parameters for the primary Extract where the mining database is a downstream database and is a multitenant container database

```
EXTRACT financep
USERIDALIAS tiger1
TRANLOGOPTIONS MININGUSERALIAS tiger2
TRANLOGOPTIONS INTEGRATEDPARAMS (MAX_SGA_SIZE 164, &
    DOWNSTREAM_REAL_TIME_MINE y)
LOGALLSUPCOLS
UPDATERECORDFORMAT COMPACT
DDL INCLUDE MAPPED SOURCECATALOG pdb1 INCLUDE MAPPED SOURCECATALOG pdb2
ENCRYPTTRAIL AES192EXTTRAIL /ggs/dirdat/lt
TABLE test.ogg.tab1;
SOURCECATALOG pdb1
SEQUENCE hr.employees_seq;
TABLE hr.*;
SOURCECATALOG pdb2
```

```
TABLE sales.*;
TABLE acct.*;
```

Parameter	Description
EXTRACT <i>group</i>	<i>group</i> is the name of the Extract group. For more information, see <i>Reference for Oracle GoldenGate</i> .
USERIDALIAS <i>alias</i>	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.
LOGALLSUPCOLS	Writes all supplementally logged columns to the trail, including those required for conflict detection and resolution and the scheduling columns required to support integrated Replicat. (Scheduling columns are primary key, unique index, and foreign key columns.) You configure the database to log these columns with GGSCI commands. See Establishing Oracle GoldenGate Credentials .
UPDATERECORDFORMAT COMPACT	Combines the before and after images of an UPDATE operation into a single record in the trail. This parameter is valid for Oracle Databases version 12c and later to support Replicat in integrated mode. Although not a required parameter, UPDATERECORDFORMAT COMPACT is a best practice and significantly improves Replicat performance.
TRANLOGOPTIONS MININGUSERALIAS <i>alias</i>	Specifies connection information for the logmining server at the downstream mining database, if being used. MININGUSERALIAS specifies the alias of the Extract user for the downstream mining database. This is the user that you created in Configuring a Downstream Mining Database . The credential for this user must be stored in the Oracle GoldenGate credential store. Use MININGUSERALIAS only if the database logmining server is in a different database from the source database; otherwise just use USERIDALIAS. When using MININGUSERALIAS, use it in addition to USERIDALIAS, because credentials are required for both databases.
TRANLOGOPTIONS [INTEGRATEDPARAMS (<i>parameter</i> [, ...])]	Optional, passes parameters to the Oracle Database that contains the database logmining server. Use only to change logmining server parameters from their default settings. See Additional Parameter Options for Integrated Capture .
TRANLOGOPTIONS CHECKPOINTRETENTIONTIME <i>days</i>	Optional, controls the number of days that Extract retains checkpoints before purging them automatically. Partial days can be specified using decimal values. For example, 8.25 specifies 8 days and 6 hours. For more information, see <i>Reference for Oracle GoldenGate</i> .
DDL <i>include_clause</i>	Required if replicating DDL operations. See Configuring DDL Support for more information.
ENCRYPTTRAIL <i>algorithm</i>	Encrypts the local trail.
EXTTRAIL <i>pathname</i>	Specifies the path name of the local trail to which the primary Extract writes captured data.

Parameter	Description
<code>SOURCECATALOG container</code>	Use this parameter when the source database is a multitenant container database. Specifies the name of a pluggable database that is to be used as the default container for all subsequent <code>TABLE</code> and <code>SEQUENCE</code> parameters that contain two-part names. This parameter enables you to use two-part object names (<i>schema.object</i>) rather than three-part names (<i>container.schema.object</i>). It remains in effect until another <code>SOURCECATALOG</code> parameter is encountered or a full three-part <code>TABLE</code> or <code>SEQUENCE</code> specification is encountered.
<code>{TABLE SEQUENCE}</code> <code>[container.]schema.object;</code>	<p>Specifies the database object for which to capture data.</p> <ul style="list-style-type: none"> <code>TABLE</code> specifies a table or a wildcarded set of tables. <code>SEQUENCE</code> specifies a sequence or a wildcarded set of sequences. <code>container</code> is the name of the pluggable database (PDB) that contains the object, if this database is a multitenant container database. The container part of the name is not required if this Extract group will only process data from one PDB and the default PDB is specified with the <code>SOURCECATALOG</code> parameter. <code>schema</code> is the schema name or a wildcarded set of schemas. <code>object</code> is the table or sequence name, or a wildcarded set of those objects. <p>Terminate the parameter statement with a semi-colon.</p> <p>To exclude a name from a wildcard specification, use the <code>CATALOGEXCLUDE</code>, <code>SCHEMAEXCLUDE</code>, <code>TABLEEXCLUDE</code>, and <code>EXCLUDEWILDCARDOBJECTSONLY</code> parameters as appropriate.</p>
<code>MAPINVISIBLECOLUMNS</code>	<p>Controls whether or not Replicat includes invisible columns in Oracle target tables for default column mapping. Configure the invisible columns in your column mapping using SQL to explicitly specify column names. For example:</p> <pre>CREATE TABLE tab1 (id NUMBER, data CLOB INVISIBLE); INSERT INTO tab1 VALUES (1, 'a');ERROR: ORA-913 INSERT INTO tab1 (id, data) VALUES (1, 'a'); OK</pre> <p>You can change the column visibility using <code>ALTER TABLE</code>. The invisible column can be part of an index, including primary key and unique index.</p>

- 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 in GGSCI.
- Save and close the file.

Configuring the Data Pump Extract

A data pump can perform data filtering, mapping, and conversion, or it can be configured in pass-through mode, where data is passively transferred as-is, without manipulation.

These steps configure the data pump that reads the local trail and sends the data across the network to a remote trail. The data pump is optional, but recommended.

Note:

If you want to perform this task using microservices, see *How to Add a Path in Using the Oracle GoldenGate Microservices Architecture*.

1. In GGSCI on the source system, create the data-pump parameter file.

```
EDIT PARAMS name
```

Where: *name* is the name of the data pump Extract.

2. Enter the data pump parameters in the order shown, starting a new line for each parameter statement. Your input variables will be different.

Basic parameters for the data pump Extract group using two-part object names from a non-CDB database:

```
EXTRACT extpump
USERIDALIAS tiger1
RMTHOST fin1, MGRPORT 7809 ENCRYPT AES192, KEYNAME securekey2
RMTRAIL /ggs/dirdat/rt
SEQUENCE hr.employees_seq;
TABLE hr.*;
```

Basic parameters for the data pump Extract group using three-part object names from a trail that contains multitenant database data (including a pluggable database):

```
EXTRACT extpump
USERIDALIAS tiger1
RMTHOST fin1, MGRPORT 7809 ENCRYPT AES192, KEYNAME securekey2
RMTRAIL /ggs/dirdat/rt
TABLE test.ogg.tab1;
SOURCECATALOG pdb1
SEQUENCE hr.employees_seq;
TABLE hr.*;
SOURCECATALOG pdb2
TABLE sales.*;
TABLE acct.*;
```

Parameter	Description
EXTRACT <i>group</i>	<i>group</i> is the name of the data pump Extract. For more information, see <i>Reference for Oracle GoldenGate</i> .
USERIDALIAS <i>alias</i>	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.
RMTHOST <i>hostname</i> , MGRPORT <i>portnumber</i> , [, ENCRYPT <i>algorithm</i> KEYNAME <i>keyname</i>]	<ul style="list-style-type: none"> • RMTHOST specifies the name or IP address of the target system. • MGRPORT specifies the port number where Manager is running on the target. • ENCRYPT specifies optional encryption of data across TCP/IP.
RMTRAIL <i>pathname</i>	Specifies the path name of the remote trail.
SOURCECATALOG <i>container</i>	Use this parameter when the source database is a multitenant container database. Specifies the name of a pluggable database that is to be used as the default container for all subsequent TABLE and SEQUENCE parameters that contain two-part names. This parameter enables you to use two-part object names (<i>schema.object</i>) rather than three-part names (<i>container.schema.object</i>). It remains in effect until another SOURCECATALOG parameter is encountered or a full three-part TABLE or SEQUENCE specification is encountered. Use this parameter when the source database is a multitenant container database.

Parameter	Description
<code>{TABLE SEQUENCE}</code> <code>[container.]schema.object;</code>	<p>Specifies a table or sequence, or multiple objects specified with a wildcard. In most cases, this listing will be the same as that in the primary Extract parameter file.</p> <ul style="list-style-type: none"> <code>TABLE</code> specifies a table or a wildcarded set of tables. <code>SEQUENCE</code> specifies a sequence or a wildcarded set of sequences. <code>container</code> is the name of the root container or pluggable database that contains the table or sequence, if this source database is a multitenant container database. See the <code>SOURCECATALOG</code> description in this table. <code>schema</code> is the schema name or a wildcarded set of schemas. <code>object</code> is the name of a table or sequence, or a wildcarded set of those objects. <p>Terminate this parameter statement with a semi-colon.</p> <p>To exclude tables or sequences from a wildcard specification, use the <code>TABLEEXCLUDE</code> or <code>SEQUENCEEXCLUDE</code> parameter after the <code>TABLE</code> statement.</p>

- 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 in GGSCI.
- Save and close the file.

Next Steps

A parameter file is a plain text file that is read by an associated Oracle GoldenGate process. Oracle GoldenGate uses two types of parameter files: a GLOBALS file and runtime parameter files.

Once you have created a basic parameter file for classic capture, see the following for related configuration steps.

[Configuring Oracle GoldenGate Apply](#)

[Configuring Oracle GoldenGate in a Multitenant Container Database](#)

[Additional Oracle GoldenGate Configuration Considerations](#)

[Configuring DDL Support](#) (to use Oracle GoldenGate DDL support)

[Creating Process Groups](#) (to use Oracle GoldenGate DDL support)

[Instantiating Oracle GoldenGate Replication](#)

[Optional Parameters for Integrated Modes](#)

[Configuring a Downstream Mining Database](#)

[Example Downstream Mining Configuration](#)

[Supporting Changes to XML Schemas](#)

7

Configuring Oracle GoldenGate Apply

This chapter contains instructions for configuring the Replicat apply process in either nonintegrated or integrated mode.

Topics:

- [Prerequisites for Configuring Replicat](#)
This topic provides the best practices for configuring Replicat.
- [What to Expect from these Instructions](#)
These instructions show you how to configure a basic Replicat parameter (configuration) file.
- [Creating a Checkpoint Table](#)
The checkpoint table is a required component of Replicat.
- [Configuring Replicat](#)
Configure a Replicat process to configure Replicat against a pluggable database. Replicat can operate in any mode within a pluggable database.
- [Next Steps](#)
Once you have created a basic parameter file for Replicat, see the following for additional configuration steps.

Prerequisites for Configuring Replicat

This topic provides the best practices for configuring Replicat.

The guidelines to follow before configuring Replicat are:

1. [Preparing the Database for Oracle GoldenGate.](#)
2. [Establishing Oracle GoldenGate Credentials.](#)
3. [Choosing Capture and Apply Modes.](#)
4. Create the Oracle GoldenGate instance on the target system by configuring the Manager process.

See *How to Add a Replicat in Using the Oracle GoldenGate Microservices Architecture.*

Note:

To switch an active Replicat configuration from one mode to the other, perform these configuration steps and then see *Administering Oracle GoldenGate.*

What to Expect from these Instructions

These instructions show you how to configure a basic Replicat parameter (configuration) file.

Your business requirements probably will require a more complex topology, but this procedure forms a basis for the rest of your configuration steps.

By performing these steps, you can:

- get the basic configuration file established.
- build upon it later by adding more parameters as you make decisions about features or requirements that apply to your environment.
- use copies of it to make the creation of additional Replicat parameter files faster than starting from scratch.

Note:

These instructions do not configure Replicat to apply DDL to the target. To support DDL, create the basic Replicat parameter file and then see [Configuring DDL Support](#) for configuration instructions.

Creating a Checkpoint Table

The checkpoint table is a required component of Replicat.

A Replicat maintains its recovery checkpoints in the checkpoint table, which is stored in the target database. Checkpoints are written to the checkpoint table within the Replicat transaction. Because a checkpoint either succeeds or fails with the transaction, Replicat ensures that a transaction is only applied once, even if there is a failure of the process or the database. See *Before Creating Replicat* in the *Using the Oracle GoldenGate Microservices Architecture* to learn to create checkpoint tables from the Microservices web UI.

Note:

This procedure installs a default checkpoint table, which is sufficient in most cases. More than one checkpoint table can be used, such as to use a different one for each Replicat group. To use a non-default checkpoint table, which overrides the default table, use the `CHECKPOINTTABLE` option of `ADD REPLICAT` when you create Replicat processes in the steps in [Instantiating Oracle GoldenGate Replication](#).

- [Adding the Checkpoint Table to the Target Database](#)
- [Specifying the Checkpoint Table in the Oracle GoldenGate Configuration](#)
- [Disabling Default Asynchronous COMMIT to Checkpoint Table](#)

Adding the Checkpoint Table to the Target Database

1. From the Oracle GoldenGate directory on the target, run GGSCI and issue the `DBLOGIN` command to log into the target database.

```
DBLOGIN USERIDALIAS alias
```

Where:

- *alias* specifies the alias of the database login credential of a user that can create tables in a schema that is accessible to Replicat. This credential must exist in the Oracle GoldenGate credential store. For more information, see [Establishing Oracle GoldenGate Credentials](#).
2. In GGSCI, create the checkpoint table in a schema of your choice (ideally dedicated to Oracle GoldenGate).

```
ADD CHECKPOINTTABLE [container.]schema.table
```

Where:

- *container* is the name of the container if *schema.table* is in a multitenant container database. This container can be the root container or a pluggable database that contains the table.
- *schema.table* are the schema and name of the table. See *Administering Oracle GoldenGate* for instructions for specifying object names.

Specifying the Checkpoint Table in the Oracle GoldenGate Configuration

To specify the checkpoint table in the Oracle GoldenGate configuration:

1. Create a `GLOBALS` file (or edit the existing one).

```
EDIT PARAMS ./GLOBALS
```

Note:

`EDIT PARAMS` creates a simple text file. When you save the file after `EDIT PARAMS`, it is saved with the name `GLOBALS` in upper case, without a file extension. It must remain as such, and the file must remain in the root Oracle GoldenGate directory.

2. In the `GLOBALS` file, enter the `CHECKPOINTTABLE` parameter.

```
CHECKPOINTTABLE [container.]schema.table
```

3. Save and close the `GLOBALS` file.

Disabling Default Asynchronous COMMIT to Checkpoint Table

When a nonintegrated Replicat uses a checkpoint table, it uses an asynchronous `COMMIT` with the `NOWAIT` option to improve performance. Replicat can continue processing immediately after applying this `COMMIT`, while the database logs the transaction in the background. You

can disable the asynchronous `COMMIT with NOWAIT` by using the `DBOPTIONS` parameter with the `DISABLECOMMITNOWAIT` option in the Replicat parameter file.

 **Note:**

When the configuration of a nonintegrated Replicat group does not include a checkpoint table, the checkpoints are maintained in a file on disk. In this case, Replicat uses `COMMIT with WAIT` to prevent inconsistencies in the event of a database failure that causes the state of the transaction, as in the checkpoint file, to be different than its state after the recovery.

Configuring Replicat

Configure a Replicat process to configure Replicat against a pluggable database. Replicat can operate in any mode within a pluggable database.

These steps configure the Replicat process.

1. In GGSCI on the target system, create the Replicat parameter file.

```
EDIT PARAMS name
```

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. See [Basic Parameters for Replicat](#) for descriptions.

Basic parameters for the Replicat group in nonintegrated mode:

```
REPLICAT financer
USERIDALIAS tiger2
ASSUMETARGETDEFS
MAP hr.*, TARGET hr2.*;
```

Basic parameters for the Replicat group in integrated Replicat mode:

```
REPLICAT financer
DBOPTIONS INTEGRATEDPARAMS(parallelism 6)
USERIDALIAS tiger2
ASSUMETARGETDEFS
MAP hr.*, TARGET hr2.*;
```

Parameter	Description
<code>REPLICAT group</code>	<i>group</i> is the name of the Replicat group.
<code>DBOPTIONS DEFERREFCONST</code>	Applies to Replicat in nonintegrated mode. <code>DEFERREFCONST</code> sets constraints to <code>DEFERRABLE</code> to delay the enforcement of cascade constraints by the target database until the Replicat transaction is committed. See <i>Reference for Oracle GoldenGate</i> for additional important information.
<code>DBOPTIONS INTEGRATEDPARAMS (parameter[, ...])</code>	This parameter specification applies to Replicat in integrated mode. It specifies optional parameters for the inbound server. See Optional Parameters for Integrated Modes for additional important information about these <code>DBOPTIONS</code> options.

Parameter	Description
USERIDALIAS <i>alias</i>	Specifies the alias of the database login credential of the user that is assigned to Replicat. This credential must exist in the Oracle GoldenGate credential store. For more information, see Establishing Oracle GoldenGate Credentials
MAP <i>[container.]schema.object,</i> TARGET <i>schema.object;</i>	<p>Specifies the relationship between a source table or sequence, or multiple objects, and the corresponding target object or objects.</p> <ul style="list-style-type: none"> 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. <i>container</i> is the name of a container, if the source database is a multitenant container database. <i>schema</i> is the schema name or a wildcarded set of schemas. <i>object</i> is the name of a table or sequence, or a wildcarded set of objects. <p>Terminate this parameter statement with a semi-colon.</p> <p>To exclude objects from a wildcard specification, use the MAPEXCLUDE parameter.</p> <p>For more information and for additional options that control data filtering, mapping, and manipulation, see MAP in <i>Reference for Oracle GoldenGate</i>.</p>

- If using integrated Replicat or parallel Replicat in integrated mode, add the following parameters to the Extract parameter file:
 - LOGALLSUPCOLS: 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 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.
- 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 in GGSCI. For more information, see the *Reference for Oracle GoldenGate* and [Optional Parameters for Integrated Modes](#) for additional configuration considerations..
- Save and close the file.

 **Note:**

See *Administering Oracle GoldenGate* for important information about making configuration changes to Replicat once processing is started, if using integrated Replicat.

Next Steps

Once you have created a basic parameter file for Replicat, see the following for additional configuration steps.

[Configuring Capture in Classic Mode](#) or [Configuring Capture in Integrated Mode](#) if you have not configured capture yet.

[Additional Configuration Steps For Using Nonintegrated Replicat](#) (if using nonintegrated Replicat)

[Additional Oracle GoldenGate Configuration Considerations](#)

[Configuring DDL Support](#) (to use Oracle GoldenGate DDL support)

[Creating Process Groups](#)

[Instantiating Oracle GoldenGate Replication](#)

8

Additional Oracle GoldenGate Configuration Considerations

This chapter contains additional configuration considerations that may apply to your database environment.

Topics:

- [Installing Support for Oracle Sequences](#)
To support Oracle sequences, you must install some database procedures.
- [Handling Special Data Types](#)
It addresses special configuration requirements for different Oracle data types
- [Handling Other Database Properties](#)
This topic describes the database properties that may affect Oracle GoldenGate and the parameters that you can use to resolve or work around the condition.
- [Controlling the Checkpoint Frequency](#)
The `CHECKPOINTRETENTIONTIME` option of the `TRANLOGOPTIONS` parameter controls the number of days that Extract in integrated mode retains checkpoints before purging them automatically.
- [Excluding Replicat Transactions](#)
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.
- [Advanced Configuration Options for Oracle GoldenGate](#)
You may need to configure Oracle GoldenGate with advanced options to suit your business needs.

Installing Support for Oracle Sequences

To support Oracle sequences, you must install some database procedures.

To Install Oracle Sequence Objects

1. In SQL*Plus, connect to the source and target Oracle systems as `SYSDBA`.
2. If you already assigned a database user to support the Oracle GoldenGate DDL replication feature, you can skip this step. Otherwise, in SQL*Plus on both systems create a database user that can also be the DDL user.

```
CREATE USER DDLuser IDENTIFIED BY password;  
GRANT CONNECT, RESOURCE, DBA TO DDLuser;
```

3. From the Oracle GoldenGate installation directory on each system, run `GGSCI`.
4. In `GGSCI`, issue the following command on each system.

```
EDIT PARAMS ./GLOBALS
```
5. In each `GLOBALS` file, enter the `GGSCHEMA` parameter and specify the schema of the DDL user that you created earlier in this procedure.

```
GGSCHEMA schema
```

6. Save and close the files.
7. In SQL*Plus on both systems, run the `sequence.sql` script from the root of the Oracle GoldenGate installation directory. This script creates some procedures for use by Oracle GoldenGate processes. (Do not run them yourself.) You are prompted for the user information that you created in the first step.

```
@sequence.sql
```

8. In SQL*Plus on the source system, grant `EXECUTE` privilege on the `updateSequence` procedure to a database user that can be used to issue the `DBLOGIN` command. Remember or record this user. You use `DBLOGIN` to log into the database prior to issuing the `FLUSH SEQUENCE` command, which calls the procedure.

```
GRANT EXECUTE on DDLuser.updateSequence TO DBLOGINuser;
```

9. In SQL*Plus on the target system, grant `EXECUTE` privilege on the `replicateSequence` procedure to the Replicat database user.

```
GRANT EXECUTE on DDLuser.replicateSequence TO Replicatuser;
```

10. In SQL*Plus on the source system, issue the following statement in SQL*Plus.

```
ALTER TABLE sys.seq$ ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;
```

To capture the sequence from a multitenant database

1. Create an Oracle GoldenGate user in each PDB that you need to capture sequences from.
2. Add the user to the `GLOBALS` parameter file. It is easier if you use the same user for each PDB, if you don't then you need to change the `GLOBALS` file each time you do step 3.
3. Run the `sequence.sql` script on each PDB using the user created in step 1.
4. Log into Admin Client or GGSCI.
5. Connect to the root container on the source using `DBLOGIN`.
6. Issue the `FLUSH SEQUENCE` command for each PDB.

If replicating sequences into a multitenant database:

1. On the target, create a user as created in step 1 in the previous section, for each PDB you are replicating sequences into.
2. Connect to the PDB using that user and run the `sequence.sql` script.

If you don't want to keep these database accounts, you can drop the user or deactivate the account.

Here is an example of the entire process:

```
Environment information
OGG 19.1 Oracle 12c to Oracle 12c Replication, Integrated
Extract, Parallel Replicat
Source: CDB GOLD, PDB CERTMISSN
Target: CDB PLAT, PDB CERTDSQ
Source Oracle GoldenGate Configuration
```

```
Container User: C##GGADMIN  
PDB User for Sequences: GGATE
```

```
sqlplus / as sysdba  
SQL> alter session set container=CERTMISSN;  
SQL> create user ggate identified by password default tablespace  
users temporary tablespace temp quota unlimited on users container=current;  
Run @sequence  
sqlplus / as sysdba  
SQL> alter session set container=CERTMISSN;  
SQL> @sequence
```

When prompted enter GGATE

```
GLOBALS  
GGSCHEMA GGATE  
Flush Sequence  
GGSCI> DBLOGIN USERIDALIAS GGADMIN DOMAIN GOLD_QC_CDB$ROOT  
GGSCI> FLUSH SEQUENCE CERTMISSN.SRCSHEMA1.  
Target OGG Configuration  
PDB User: GGATE  
Run @sequence  
sqlplus / as sysdba  
SQL> alter session set container=CERTDSQ;  
SQL> @sequence
```

When prompted enter GGATE.

Handling Special Data Types

It addresses special configuration requirements for different Oracle data types

This section applies whether Extract operates in classic or integrated capture mode, unless otherwise noted.

- [Multibyte Character Types](#)
- [Oracle Spatial Objects](#)
- [TIMESTAMP](#)
- [Large Objects \(LOB\)](#)
- [XML](#)
- [User Defined Types](#)

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.

Oracle Spatial Objects

To replicate tables that contain one or more columns of `SDO_GEORASTER` object type from an Oracle source to an Oracle target, follow these instructions to configure Oracle GoldenGate to process them correctly.

1. Create a `TABLE` statement and a `MAP` statement for the georaster tables and also for the related raster data tables.
2. If the `METADATA` attribute of the `SDO_GEORASTER` data type in any of the values exceeds 1 MB, use the `DBOPTIONS` parameter with the `XMLBUFSIZE` option to increase the size of the memory buffer that stores the embedded `SYS.XMLTYPE` attribute of the `SDO_GEORASTER` data type. If the buffer is too small, Extract abends. See `XMLBUFSIZE` in *Reference for Oracle GoldenGate*.
3. To ensure the integrity of the target georaster tables and the spatial data, keep the trigger enabled on both source and target. Use the `REPERROR` option of the `MAP` parameter to handle the "ORA-01403 No data found" error that occurs as a result of keeping the trigger enabled on the target. It occurs when a row in the source georaster table is deleted, and the trigger cascades the delete to the raster data table. Both deletes are replicated. The replicated parent delete triggers the cascaded (child) delete on the target. When the replicated child delete arrives, it is redundant and generates the error. To use `REPERROR`, do the following:
 - Use a `REPERROR` statement in each `MAP` statement that contains a raster data table.
 - Use Oracle error 1403 as the SQL error.
 - Use any of the response options as the error handling.

A sufficient way to handle the errors on raster tables caused by active triggers on target georaster tables is to use `REPERROR` with `DISCARD` to discard the cascaded delete that triggers them. The trigger on the target georaster table performs the delete to the raster data table, so the replicated one is not needed.

```
MAP geo.st_rdt, TARGET geo.st_rdt, REPERROR (-1403, DISCARD) ;
```

If you need to keep an audit trail of the error handling, use `REPERROR` with `EXCEPTION` to invoke exceptions handling. For this, you create an exceptions table and map the source raster data table twice:

- once to the actual target raster data table (with `REPERROR` handling the 1403 errors).
- again to the exceptions table, which captures the 1403 error and other relevant information by means of a `COLMAP` clause.

For more information about using an exceptions table, see *Administering Oracle GoldenGate for Windows and UNIX*.

For more information about `REPERROR` options, see *Reference for Oracle GoldenGate*.

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 LOBs in both classic capture and integrated capture mode.

1. Store large objects out of row if possible.
2. (Applies only to integrated capture) Integrated capture 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 both classic and integrated capture modes do 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](#).
- (Applies only to integrated capture) Integrated capture 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.

User Defined Types

If Oracle Database is compatible with releases greater than or equal to 12.0.0.0.0, then integrated 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.

Handling Other Database Properties

This topic describes the database properties that may affect Oracle GoldenGate and the parameters that you can use to resolve or work around the condition.

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

Database Property	Concern/Resolution
Table with interval partitioning	To support tables with interval partitioning, make certain that the <code>WILDCARDRESOLVE</code> parameter remains at its default of <code>DYNAMIC</code> .
Table with virtual columns	Virtual columns are not logged, and Oracle does not permit DML on virtual columns. You can, however, capture this data and map it to a target column that is not a virtual column by doing the following: Include the table in the Extract <code>TABLE</code> statement and use the <code>FETCHCOLS</code> option of <code>TABLE</code> to fetch the value from the virtual column in the database. In the Replicat <code>MAP</code> statement, map the source virtual column to the non-virtual target column.
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 <code>KEYCOLS</code> clause in the same <code>MAP</code> statement in which the associated source and target tables are mapped.
Redo logs or archives in different locations	The <code>TRANLOGOPTIONS</code> 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. These options may be required when Extract operates in classic capture mode. For more information, see <i>Reference for Oracle GoldenGate</i> .

Database Property	Concern/Resolution
TRUNCATE operations	<p>To replicate TRUNCATE operations, choose one of two options:</p> <ul style="list-style-type: none"> • 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. To install this support, see Installing Trigger-Based DDL Capture..
Sequences	<p>To replicate DDL for sequences (CREATE, ALTER, DROP, RENAME), use Oracle GoldenGate DDL support.</p> <p>To replicate just sequence values, use the SEQUENCE parameter in the Extract parameter file. This does <i>not</i> require the Oracle GoldenGate DDL support environment. For more information, see <i>Reference for Oracle GoldenGate</i>.</p>

Controlling the Checkpoint Frequency

The CHECKPOINTRETENTIONTIME option of the TRANLOGOPTIONS parameter controls the number of days that Extract in integrated mode 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. For more information about this parameter, see *Reference for Oracle GoldenGate*.

Excluding Replicat Transactions

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.

There are two methods to accomplish this as follows:

Method 1

Valid only for Oracle to Oracle implementations.

When Extract is in classic or integrated mode (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 a classic or integrated 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.

For Oracle to Oracle, this is the recommended method.

Method 2

Valid for any implementation; Oracle or heterogeneous database configurations.

Alternatively, when Extract is in classic or integrated capture mode, you could also use the Extract `TRANLOGOPTIONS` parameter with the `EXCLUDEUSER` or `EXCLUDEUSERID` option to ignore Replicat the 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.

For more information, see *Reference for Oracle GoldenGate*.

Advanced Configuration Options for Oracle GoldenGate

You may need to configure Oracle GoldenGate with advanced options to suit your business needs.

See the following:

- For additional configuration guidelines to achieve specific replication topologies, see *Administering Oracle GoldenGate*. This guide includes instructions for the following configurations:
 - Using Oracle GoldenGate for live reporting
 - Using Oracle GoldenGate for real-time data distribution
 - Configuring Oracle GoldenGate for real-time data warehousing
 - Using Oracle GoldenGate to maintain a live standby database
 - Using Oracle GoldenGate for active-active high availability

That guide also contains information about:

- Oracle GoldenGate architecture
 - Oracle GoldenGate commands
 - Oracle GoldenGate initial load methods
 - Configuring security
 - Using customization features
 - Configuring data filtering and manipulation
- If either the source or target database is non-Oracle, follow the installation and configuration instructions in the Oracle GoldenGate installation and setup guide for that database, and then refer to the Oracle GoldenGate administration and reference documentation for further information.

9

Additional Configuration Steps For Using Nonintegrated Replicat

This chapter contains instructions that are specific only to Replicat when operating in *nonintegrated* mode. When Replicat operates in nonintegrated mode, triggers, cascade constraints, and unique identifiers must be properly configured in an Oracle GoldenGate environment.

This chapter is a supplement to the basic configuration requirements that are documented in [Configuring Oracle GoldenGate Apply](#).

Topics:

- [Disabling Triggers and Referential Cascade Constraints on Target Tables](#)
Triggers and cascade constraints must be disabled on Oracle target tables when Replicat is in nonintegrated mode.

Disabling Triggers and Referential Cascade Constraints on Target Tables

Triggers and cascade constraints must be disabled on Oracle target tables when Replicat is in nonintegrated mode.

Constraints must be disabled in nonintegrated Replicat mode because Oracle GoldenGate replicates DML that results from the firing of a trigger or a cascade constraint. If the same trigger or constraint gets activated on the target table, it becomes redundant because of the replicated version, and the database returns an error. Consider the following example, where the source tables are `emp_src` and `salary_src` and the target tables are `emp_targ` and `salary_targ`.

1. A delete is issued for `emp_src`.
2. It cascades a delete to `salary_src`.
3. Oracle GoldenGate sends both deletes to the target.
4. The parent delete arrives first and is applied to `emp_targ`.
5. The parent delete cascades a delete to `salary_targ`.
6. The cascaded delete from `salary_src` is applied to `salary_targ`.
7. The row cannot be located because it was already deleted in step 5.

10

Configuring DDL Support

This chapter contains information to help you understand and configure DDL support in Oracle GoldenGate.

Topics:

- [Prerequisites for Configuring DDL](#)
Extract can capture DDL operations from a source Oracle Database through the use of a special DDL trigger or natively through the Oracle logmining server.
- [Overview of DDL Synchronization](#)
Oracle GoldenGate supports the synchronization of DDL operations from one database to another.
- [Limitations of Oracle GoldenGate DDL Support](#)
This topic contains some limitations of the DDL feature.
- [Configuration Guidelines for DDL Support](#)
The following are guidelines to take into account when configuring Oracle GoldenGate processes to support DDL replication.
- [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.
- [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.
- [Enabling DDL Support](#)
Data Definition Language (DDL) is useful in dynamic environments which change constantly.
- [Filtering DDL Replication](#)
By default, all DDL is passed to Extract.
- [Special Filter Cases](#)
This topic describes the special cases that you must consider before creating your DDL filters.
- [How Oracle GoldenGate Handles Derived Object Names](#)
DDL operations can contain a *base object* name and also a *derived object* name.
- [Using DDL String Substitution](#)
You can substitute strings within a DDL operation while it is being processed by Oracle GoldenGate.
- [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.
- [Adding Supplemental Log Groups Automatically](#)
Use the `DDLOPTIONS` parameter with the `ADDTRANADATA` option for performing tasks described in this topic.

- [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.
- [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.
- [How DDL is Evaluated for Processing](#)
This topic explains how Oracle GoldenGate processes DDL statements on the source and target systems.
- [Viewing DDL Report Information](#)
By default, Oracle GoldenGate shows basic statistics about DDL at the end of the Extract and Replicat reports.
- [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 Tools that Support Trigger-Based DDL Capture](#)
This section documents the additional tools available to support trigger-based capture.
- [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.

Prerequisites for Configuring DDL

Extract can capture DDL operations from a source Oracle Database through the use of a special DDL trigger or natively through the Oracle logmining server.

Which of these methods you can use depends on the Extract capture mode and the version of the source Oracle Database. This section describes the available support in each capture mode, see [Choosing Capture and Apply Modes](#).

- [Support for DDL Capture in Integrated Capture Mode](#)
- [Support for DDL Capture in Classic Capture Mode](#)

Support for DDL Capture in Integrated Capture Mode

The integrated capture mode of Extract supports two DDL capture methods:

- **Oracle 11.2.0.4 or later:** Oracle Databases that have the database `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* (also known as *triggerless DDL capture*). No trigger or installed supportive objects are required. 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.

- **Versions earlier than 11.2.0.4:** Oracle Databases that have the `COMPATIBLE` parameter set to anything earlier than 11.2.0.4 require the use of the Oracle GoldenGate DDL trigger. To use trigger-based DDL capture, you must install the DDL trigger and supporting database objects before you configure Extract for DDL support.

Support for DDL Capture in Classic Capture Mode

Classic capture mode requires the use of the Oracle GoldenGate DDL trigger to capture DDL from an Oracle Database. Native DDL capture is not supported by classic capture mode.

DDL capture from a multitenant container database is not supported by classic capture mode.

When you are using Classic capture mode and replicating a `CREATE USER` using the DDL trigger, the trigger owner and the Extract login user *must* match to avoid a privilege error when attempting to replicate the `CREATE USER` command.

To use trigger-based DDL capture, you must install the DDL trigger and supporting database objects before you configure Extract for DDL support, see [Installing Trigger-Based DDL Capture](#).

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

Limitations of Oracle GoldenGate DDL Support

This topic contains some limitations of the DDL feature.

For any additional limitations that were found after this documentation was published, see the *Release Notes for Oracle GoldenGate*.

- [DDL Statement Length](#)
- [Supported Topologies](#)
- [Filtering, Mapping, and Transformation](#)
- [Renames](#)
- [Interactions Between Fetches from a Table and DDL](#)
- [Comments in SQL](#)
- [Compilation Errors](#)

- [Interval Partitioning](#)
- [DML or DDL Performed Inside a DDL Trigger](#)
- [LogMiner Data Dictionary Maintenance](#)

DDL Statement Length

Oracle GoldenGate measures the length of a DDL statement in bytes, not in characters. The supported length is approximately 4 MB, allowing for some internal overhead that can vary in size depending on the name of the affected object and its DDL type, among other characteristics. If the DDL is longer than the supported size, Extract will issue a warning and ignore the DDL operation.

If Extract is capturing DDL by means of the DDL trigger, the ignored DDL is saved in the marker table. You can capture Oracle DDL statements that are ignored, as well as any other Oracle DDL statement, by using the `ddl_ddl2file.sql` script, which saves the DDL operation to a text file in the `USER_DUMP_DEST` directory of Oracle. The script prompts for the following input:

- The name of the schema that contains the Oracle GoldenGate DDL objects, which is specified in the `GLOBALS` file.
- The Oracle GoldenGate marker sequence number, which is recorded in the Extract report file when `DDLOPTIONS` with the `REPORT` option is used in the Extract parameter file.
- A name for the output file.

Supported Topologies

Oracle GoldenGate supports DDL synchronization only in a like-to-like configuration. The source and target object definitions must be identical.

DDL replication is only supported for Oracle to Oracle replication. It is not supported between different databases, like Oracle to Teradata, or SQL Server to Oracle.

Oracle GoldenGate does not support DDL on a standby database.

Oracle GoldenGate supports DDL replication in all supported unidirectional configurations, and in bidirectional configurations between two, and only two, systems. 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. Mapping or filtering of DDL by a data-pump Extract is not permitted, and the DDL is passed as it was received from the primary Extract.

For example, `ALTER TABLE TableA` is processed by a data pump as `ALTER TABLE TableA`. It cannot be mapped by that process as `ALTER TABLE TableB`, regardless of any `TABLE` statements that specify otherwise.

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 tab1 TO tab2` could be changed to `ALTER TABLE schema.tab1 RENAME TO schema.tab2`. 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, in classic capture, partial updates on LOBs (modifications done via `dbms_lob` package) 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 (both in classic and integrated capture).

 **Note:**

Integrated capture 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 in GGSCI 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 t2 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

The following are guidelines to take into account when configuring Oracle GoldenGate processes to support DDL replication.

- [Database Privileges](#)
- [Parallel Processing](#)

- [Object Names](#)
- [Data Definitions](#)
- [Truncates](#)
- [Initial Synchronization](#)
- [Data Continuity After CREATE or RENAME](#)

Database Privileges

For database privileges that are required for Oracle GoldenGate to support DDL capture and replication, see [Establishing Oracle GoldenGate Credentials](#).

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 `ReplicatA` processes DML for `Table1`, then it should also process the DDL for `Table1`. If `Table2` has a foreign key to `Table1`, then its DML and DDL operations also should be processed by `ReplicatA`.

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 *Reference for Oracle GoldenGate*.

For more information about using a definitions file, see *Administering 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 *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 new 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.

- [Mapped Scope](#)
- [Unmapped Scope](#)
- [Other Scope](#)

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.
- See [Understanding DDL Scopes](#).

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;
```

See [Understanding DDL Scopes](#).

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 SCOTT/TIGER
CREATE TABLE TAB1 (X NUMBER);
CREATE TABLE SRC1.TAB2(X NUMBER) AS SELECT * FROM TAB1;
```

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 SCOTT/TIGER
ALTER SESSION SET CURRENT_SCHEMA=SRC;
CREATE TABLE TAB1 (X NUMBER);
CREATE TABLE SRC1.TAB2(X NUMBER) AS SELECT * FROM TAB1;
```

In both of those DDL statements, the unqualified table `TAB1` is resolved as `SRC.TAB1` based on the current schema `SRC` that is in effect during the DDL execution.

In both classic and integrated capture modes, 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 following methods to filter DDL operations so that specific (or all) DDL is applied to the target database according to your requirements.

- **Filtering with PL/SQL Code:** Valid only for trigger-based DDL capture. This method makes use of an Oracle function that is called by the DDL trigger when a DDL operation occurs, to compute whether or not to send the DDL to Extract. Filtering with PL/SQL code should only be used to improve the performance of the source database when the DDL trigger is in use. It can be combined with built-in rules and DDL parameter filtering (see the following). Any DDL that is passed to Extract after it is filtered by the DDL trigger or filter rules can be filtered further with the `DDL` parameter to meet specific needs.
- **Filtering with Built-In Filter Rules:** Valid only for trigger-based DDL capture. This method makes use of some procedures that you run to build filter rules into the Oracle GoldenGate trigger logic. This method allows discreet control over the types of objects that are sent to Extract, and it allows the ordering of rule evaluation. This method should only be used to improve the performance of the source database when the DDL trigger is in use. You can combine built-in rules with PL/SQL and DDL parameter filtering. Any DDL that is passed to Extract after it is filtered by the DDL trigger or filter rules can be filtered further with the `DDL` parameter to meet specific needs.

 **Note:**

Filtering with PL/SQL or built-in filter rules is unnecessary for an Extract that operates in integrated-capture mode. If Extract must operate in classic mode and you use these filtering methods, the same filtering must happen for any transactional data (DML) that is associated with the filtered objects. For example, if you filter out the DDL that creates a table named `ACCOUNTS`, make certain the `ACCOUNTS` table is not specified in any `TABLE` or `MAP` statements, or use the appropriate exclusion parameter to exclude it from wildcard resolution. See *Reference for Oracle GoldenGate* for a list of wildcard exclusion parameters.

- **Filtering with DDL Parameter:** Valid for both trigger-based and 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 all of the 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 than the trigger method, including the ability to filter collectively based on the DDL scope (for example, include all `MAPPED` scope).

 **Note:**

If a DDL operation fails in the middle of a `TRANSACTION`, it forces a commit, which means that the transaction spanning the DDL is split into two. The first half is committed and the second half can be restarted. If a recovery occurs, the second half of the transaction cannot be filtered since the information contained in the header of the transaction is no longer there.

- [Filtering with PL/SQL Code](#)
- [Filtering With Built-in Filter Rules](#)
- [Filtering with the DDL Parameter](#)

Filtering with PL/SQL Code

This method is only valid for trigger-based capture.

You can write PL/SQL code to pass information about the DDL to a function that computes whether or not the DDL is passed to Extract. By sending fewer DDL operations to Extract, you can improve capture performance.

1. Copy the `ddl_filter.sql` file that is in the Oracle GoldenGate installation directory to a test machine where you can test the code that you will be writing.
2. Open the file for editing. It contains a PL/SQL function named `filterDDL`, which you can modify to specify `if/then` filter criteria. The information that is passed to this function includes:
 - `ora_owner`: the schema of the DDL object
 - `ora_name`: the defined name of the object

- `ora_objtype`: the type of object, such as `TABLE` or `INDEX`
- `ora_optype`: the operation type, such as `CREATE` or `ALTER`
- `ora_login_user`: The user that executed the DDL
- `retVal`: can be either `INCLUDE` to include the DDL, or `EXCLUDE` to exclude the DDL from Extract processing.

In the location after the `'compute retVal here'` comment, write filter code for each type of DDL that you want to be filtered. The following is an example:

```
if ora_owner='SYS' then
retVal:='EXCLUDE';
end if;
if ora_objtype='USER' and ora_optype ='DROP' then
retVal:='EXCLUDE';
end if;
if ora_owner='JOE' and ora_name like 'TEMP%' then
retVal:='EXCLUDE';
end if;
```

In this example, the following DDL is excluded from being processed by the DDL trigger:

- DDL for objects owned by `SYS`
- any `DROP USER`
- any DDL on `JOE.TEMP%`

3. (Optional) To trace the filtering, you can add the following syntax to each `if/then` statement in the PL/SQL:

```
if ora_owner='JOE' and ora_name like 'TEMP%' then
retVal:='EXCLUDE';
if "&gg_user" .DDLReplication.trace_level >= 1 then
"&gg_user" .trace_put_line ('DDLFILTER', 'excluded JOE.TEMP%');
end if;
```

Where:

- `&gg_user` is the schema of the Oracle GoldenGate DDL support objects.
- `.DDLReplication.trace_level` is the level of DDL tracing. To use trigger tracing, the `TRACE` or `TRACE2` parameter must be used with the `DDL` or `DDLONLY` option in the Extract parameter file. The `.DDLReplication.trace_level` parameter must be set to `>=1`.
- `trace_put_line` is a user-defined text string that Extract writes to the trace file that represents the type of DDL that was filtered.

4. Save the code.
5. Stop DDL activity on the test system.
6. In SQL*Plus, compile the `ddl_filter.sql` file as follows, where `schema_name` is the schema where the Oracle GoldenGate DDL objects are installed.

```
@ddl_filter schema_name
```


7. Test in the test environment to make certain that the filtering works. It is important to perform this testing, because any errors in the code could cause source and target DDL to become out of synchronization.
8. After a successful test, copy the file to the Oracle GoldenGate installation directory on the source production system.
9. Stop DDL activity on the source system.
10. Compile the `ddl_filter.sql` file as you did before.

```
@ddl_filter schema_name
```
11. Resume DDL activity on the source system.

Filtering With Built-in Filter Rules

This method is only valid for trigger-based capture.

You can add inclusion and exclusion rules to control the DDL operations that are sent to Extract by the DDL trigger. By storing rules and sending fewer DDL operations to Extract, you can improve capture performance.

1. Use the `DDLAUX.addRule()` function to define your rules according to the following instructions. This function is installed in the Oracle GoldenGate DDL schema after the DDL objects are installed with the `ddl_setup.sql` script.
2. To activate the rules, execute the function in SQL*Plus or enter a collection of rules in a SQL file and execute that file in SQL*Plus.

- [DDLAUX.addRule\(\) Function Definition](#)
- [Parameters for DDLAUX.addRule\(\)](#)
- [Valid DDL Components for DDLAUX.addRule\(\)](#)
- [Examples of Rule-based Trigger Filtering](#)
- [Dropping Filter Rules](#)

DDLAUX.addRule() Function Definition

```
FUNCTION addRule( obj_name IN VARCHAR2 DEFAULT NULL,  
base_obj_name IN VARCHAR2 DEFAULT NULL,  
owner_name IN VARCHAR2 DEFAULT NULL,  
base_owner_name IN VARCHAR2 DEFAULT NULL,  
base_obj_property IN NUMBER DEFAULT NULL,  
obj_type IN NUMBER DEFAULT NULL,  
command IN VARCHAR2 DEFAULT NULL,  
inclusion IN boolean DEFAULT NULL ,  
sno IN NUMBER DEFAULT NULL)  
RETURN NUMBER;
```

Parameters for DDLAUX.addRule()

The information passed to this function are the following parameters, which correlate to the attributes of an object. All parameters are optional, and more than one parameter can be specified.

- `sno`: Specifies a serial number that identifies the rule. The order of evaluation of rules is from the lowest serial number to the highest serial number, until a match is found. The

`sno` can be used to place inclusion rules ahead of an exclusion rule, so as to make an exception to the exclusion rule. Because this is a function and not a procedure, it returns the serial number of the rule, which should be used for the drop rule specified with `DDL AUX.dropRule()`. The serial number is generated automatically unless you specify one with this statement at the beginning of your code: `DECLARE sno NUMBER; BEGIN sno :=`

For example:

```
DECLARE
  sno NUMBER;
BEGIN
  sno := tkggadmin..DDL AUX.ADDRULE(obj_name => 'GGS%' ,
                                   obj_type => TYPE_TABLE);
END;
/
```

- `obj_name`: Specifies the object name. If the name is case-sensitive, enclose it within double quotes.
- `owner_name`: Specifies the name of the object schema
- `base_obj_name`: Specifies the base object name of the DDL object (such as the base table if the object is an index). If the name is case-sensitive, enclose it within double quotes.
- `base_owner_name`: Specifies the base object schema name.
- `base_obj_property`: Specifies the base object property.
- `obj_type`: Specifies the object type.
- `command`: Specifies the command.
- `inclusion = TRUE`: Indicates that the specified objects are to be captured by the DDL trigger. If this parameter is not specified, the rule becomes an exclusion rule, and the specified objects are not captured. You can specify both an exclusion rule and an inclusion rule. If a DDL does not match any of the rules, it is included (passed to Extract) by default. Calling `DDL AUX.addRule()` without any parameters generates an *empty rule* that excludes all DDL on all the objects.

Valid DDL Components for DDLAUX.addRule()

The following are the defined DDL object types, base object properties, and DDL commands that can be specified in the function code.

Valid object types are:

```
TYPE_INDEX
TYPE_TABLE
TYPE_VIEW
TYPE_SYNONYM
TYPE_SEQUENCE
TYPE_PROCEDURE
TYPE_FUNCTION
TYPE_PACKAGE
TYPE_TRIGGER
```

Valid base object properties are:

```
TB_IOT
TB_CLUSTER
TB_NESTED
TB_TEMP
TB_EXTERNAL
```

Valid commands are:

```
CMD_CREATE
CMD_DROP
CMD_TRUNCATE
CMD_ALTER
```

Examples of Rule-based Trigger Filtering

The following example excludes all temporary tables, except tables with names that start with IMPTEMP.

1. `DDLAX.ADDRULE(obj_name => 'IMPTEMP%', base_obj_property => TB_TEMP, obj_type => TYPE_TABLE, INCLUSION => TRUE);`
2. `DDLAX.ADDRULE(base_obj_property => TB_TEMP, obj_type => TYPE_TABLE);`



Note:

Since the IMPTEMP% tables must be included, that rule should come first.

The following example excludes all tables with name 'GGS%'

```
DECLARE sno NUMBER; BEGIN sno := DDLAX.ADDRULE(obj_name => 'GGS%' , obj_type =>
TYPE_TABLE); END
```

The following example excludes all temporary tables.

```
DDLAX.ADDRULE(base_obj_property => TB_TEMP, obj_type => TYPE_TABLE);
```

The following example excludes all indexes on TEMP tables.

```
DDLAX.ADDRULE(base_obj_property => TB_TEMP, obj_type => TYPE_INDEX);
```

The following example excludes all objects in schema TKGGADMIN.

```
DDLAX.ADDRULE(owner_name => 'TKGGADMIN');
```

The following example excludes all objects in TRUNCATE operations made to TEMP tables.

```
DDLAX.ADDRULE(base_obj_property => TB_TEMP, obj_type => TYPE_TABLE, command =>
CMD_TRUNCATE)
```

Dropping Filter Rules

Use the `DDLAX.dropRule()` function with the drop rule. This function is installed in the Oracle GoldenGate DDL schema after the DDL objects are installed with the `ddl_setup.sql` script. As input, specify the serial number of the rule that you want to drop.

```
FUNCTION dropRule(sno IN NUMBER) RETURN BOOLEAN;
```

Filtering with the DDL Parameter

This method is valid for trigger-based and integrated capture modes.

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, but not for a data-pump Extract.
- When combined, multiple filter option specifications are linked logically as `AND` statements.
- All filter criteria specified with multiple options must be satisfied for a DDL statement to be replicated.
- When using complex DDL filtering criteria, it is recommended that you test your configuration in a test environment before using it in production.

For `DDL` parameter syntax and additional usage guidelines, see *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](#)
- [Implicit DDL](#)

DDL EXCLUDE ALL

`DDL EXCLUDE ALL` is a special processing option that is intended primarily for Extract when using trigger-based DDL capture. `DDL EXCLUDE ALL` blocks the replication of DDL operations, but ensures that Oracle GoldenGate continues to keep the object metadata current. When Extract receives DDL directly from the logging server (triggerless DDL capture mode), current metadata is always maintained.

You can use `DDL EXCLUDE ALL` when using a method other than Oracle GoldenGate to apply DDL to the target and you want Oracle GoldenGate to replicate data changes to the target objects. It provides the current metadata to Oracle GoldenGate as objects change, thus preventing the need to stop and start the Oracle GoldenGate processes. The following special conditions apply to `DDL EXCLUDE ALL`:

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

To prevent all DDL metadata and operations from being replicated, omit the `DDL` parameter entirely.

Implicit DDL

User-generated DDL operations can generate implicit DDL operations. For example, the following statement generates two distinct DDL operations.

```
CREATE TABLE customers (custID number, name varchar2(50), address varchar2(75),  
address2 varchar2(75), city varchar2(50), state (varchar2(2), zip number, contact  
varchar2(50), areacode number(3), phone number(7), primary key (custID));
```

The first (explicit) DDL operation is the `CREATE TABLE` statement itself.

The second DDL operation is an implicit `CREATE UNIQUE INDEX` statement that creates the index for the primary key. This operation is generated by the database engine, not a user application.

Guidelines for Filtering Implicit DDL

How to filter implicit DDL depends on the mechanism that you are using to filter DDL. See [Filtering DDL Replication](#) for more information.

- When the `DDL` parameter is used to filter DDL operations, Oracle GoldenGate filters out any implicit DDL by default, because the explicit DDL will generate the implicit DDL on

the target. For example, the target database will create the appropriate index when the `CREATE TABLE` statement in the preceding example is applied by Replicat.

- When the DDL trigger is being used to filter DDL operations, you must handle the implicit DDL in your filter rules based on the following:
 - If your filtering rules exclude the explicit DDL from being propagated, you must also create a rule to exclude the implicit DDL. For example, if you exclude the `CREATE TABLE` statement in the following example, but do not exclude the implicit `CREATE UNIQUE INDEX` statement, the target database will try to create the index on a non-existent table.

```
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));
```
 - If your filtering rules permit the propagation of the explicit DDL, you do not need to exclude the implicit DDL. It will be handled correctly by Oracle GoldenGate and the target database.

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 Object, But Not Derived Object](#)
- [MAP Exists for Base and Derived Objects](#)
- [MAP Exists for Derived Object, But Not Base Object](#)
- [New Tables as Derived Objects](#)
- [Disabling the Mapping of Derived Objects](#)

MAP Exists for Base Object, But Not Derived Object

If there is a `MAP` statement for the base object, but not for the derived object, the result is a schema based on the mapping that matches the derived object name. Derived

objects are only mapped if the `MAPDERIVED` option is enabled, which is also the default option.

For example, consider the following:

Extract (source)

```
Table hr.*;
```

Replicat (target)

```
MAP hr.*, TARGET hrBackup.*;
```

Assume the following source DDL statement:

```
CREATE INDEX hr.indexPayrollDate ON TABLE hr.Payroll (payDate);
```

The `CREATE INDEX` statement is executed by Replicat on the target as follows:

```
CREATE INDEX hrBackup.indexPayrollDate ON TABLE hrBackup.Payroll (payDate);
```

In this example, the mapping is such that it matches the derived object name because of which the derived object schema is changed from `hr` to `hrBackup`.

Here's another example, where there is no mapping that matches the derived object name so the derived object name remains the same.

Extract (source)

```
Table hr.tab*;
```

Replicat (target)

```
MAP hr.tab*, TARGET hrBackup.*;
```

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 hr.indexPayrollDate ON TABLE hrBackup.tabPayroll (payDate);
```

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`
- [CREATE TABLE AS SELECT](#)
- [RENAME and ALTER TABLE RENAME](#)

CREATE TABLE AS SELECT

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.

Base Object	Derived Object	MAP/NOMAP DERIVED?	Derived object converted per a MAP?	Derived object gets schema of base object?
mapped ¹	mapped	MAPDERIVED	yes	no
mapped	not mapped	MAPDERIVED	no	yes
not mapped	mapped	MAPDERIVED	no	no
not mapped	not mapped	MAPDERIVED	no	no
mapped	mapped	NOMAPDERIVED	no	no
mapped	not mapped	NOMAPDERIVED	no	no
not mapped	mapped	NOMAPDERIVED	no	no
not mapped	not mapped	NOMAPDERIVED	no	no

¹ Mapped means included in a `MAP` statement.

The following examples illustrate the results of `MAPDERIVED` as compared to `NOMAPDERIVED`. In the following table, both trigger and table are owned by `rpt` on the target because both base and derived names are converted by means of `MAPDERIVED`.

MAP statement	Source DDL statement captured by Extract	Target DDL statement applied by Replicat
<code>MAP fin.*, TARGET rpt.*;</code>	<code>CREATE TRIGGER fin.act_trig ON fin.acct;</code>	<code>CREATE TRIGGER rpt.act_trig ON rpt.acct;</code>

In the following table, the trigger is owned by `fin`, because conversion is prevented by means of `NOMAPDERIVED`.

MAP statement	Source DDL statement captured by Extract	Target DDL statement applied by Replicat
<code>MAP fin.*, TARGET rpt.*;</code>	<code>CREATE TRIGGER fin.act_trig ON fin.acct;</code>	<code>CREATE TRIGGER fin.act_trig ON rpt.acct;</code>

 **Note:**

In the case of a `RENAME` statement, the new table name is considered to be the base table name, and the old table name is considered to be the derived table name.

Using DDL String Substitution

You can substitute strings within a DDL operation while it is being processed by Oracle GoldenGate.

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 *Reference for Oracle GoldenGate*.



Note:

Before you create a `DDLSUBST` parameter statement, it might help to review [How DDL is Evaluated for Processing](#) in this chapter.

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](#)
- [Propagating DDL in a Cascading Configuration](#)

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)

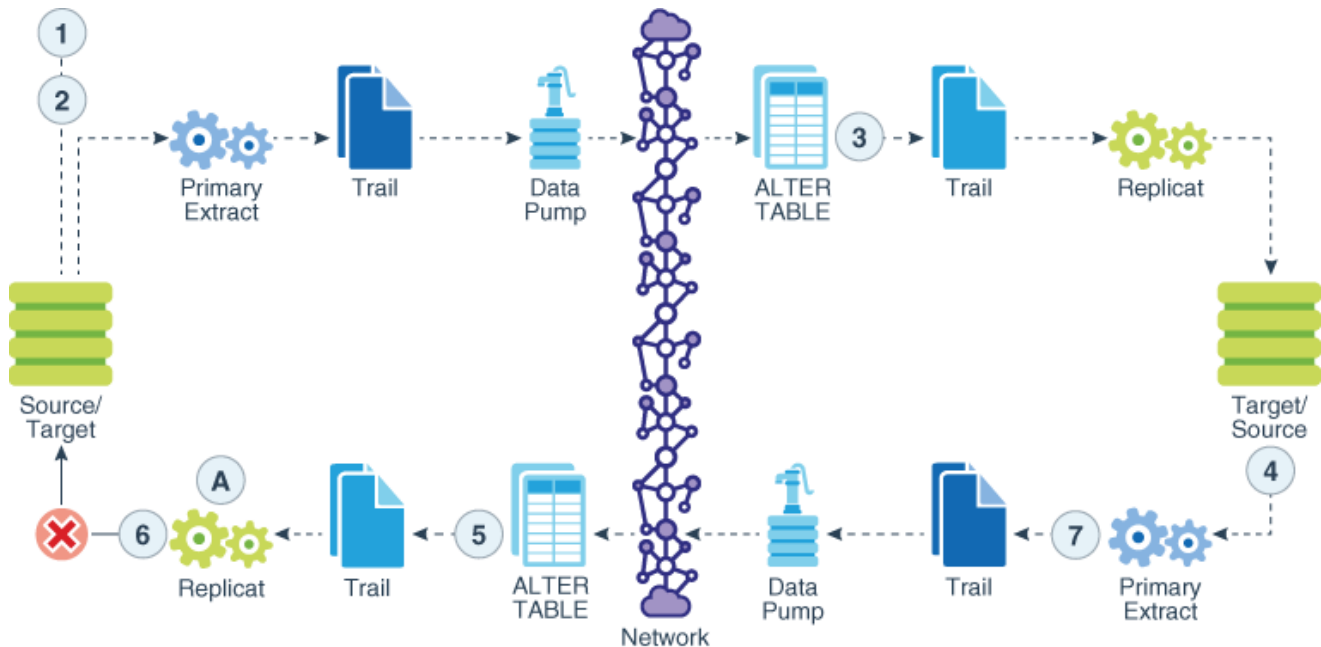
```
DDLOPTIONS GETREPLICATES, GETAPPLOPS
```

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.



The labels in the diagrams imply the following:

- 1: ALTER TABLE Customer ADD Birth_Date date
- 2; New metadata: First_Name varchar2(50), Last_Name varchar2(50), Address varchar2(50), City varchar2(50), Country varchar2(25), Birth_Date date
- 3: ALTER TABLE
- 4: New metadata: First_Name varchar2(50), Last_Name varchar2(50), Address varchar2(50), City varchar2(50), Country varchar2(25), Birth_Date date
- 5: ALTER TABLE
- 6: DDLOPTIONS UPDATEMETADATA New metadata: First_Name varchar2(50), Last_Name varchar2(50), Address varchar2(50), City varchar2(50), Country varchar2(25), Birth_Date date

For more information about `DDLOPTIONS`, see *Reference for Oracle GoldenGate*.

For more information about configuring a bidirectional configuration, see *Administering Oracle GoldenGate*.

Propagating DDL in a Cascading Configuration

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` in *Reference for Oracle GoldenGate*.

Adding 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 in GGSCI 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 [Configuring Logging Properties](#) for more information.

Removing Comments from Replicated DDL

You can use the `DDLOPTIONS` parameter with the `REMOVECOMMENTS BEFORE` and `REMOVECOMMENTS AFTER` options to prevent comments that were used in the source DDL from being included in the target DDL.

By default, comments are not removed, so that they can be used for string substitution.

Replicating an IDENTIFIED BY Password

Use the `DDLOPTIONS` parameter with the `DEFAULTUSERPASSWORDALIAS` and `REPLICATEPASSWORD | NOREPLICATEPASSWORD` options to control how the password of a

replicated {CREATE | ALTER} USER *name* IDENTIFIED BY *password* statement is handled. These options must be used together.

See the `USEPASSWORDVERIFIERLEVEL` option of `DDLOPTIONS` for important information about specifying the password verifier when Replicat operates against an Oracle 10g or 11g database.

 **Note:**

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

How DDL is Evaluated for Processing

This topic explains how Oracle GoldenGate processes DDL statements on the source and target systems.

It shows the order in which different criteria in the Oracle GoldenGate parameters are processed, and it explains 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 `DDLSUBST` 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 ADDTRANADATA`. 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.

Replicat

1. Replicat reads the DDL statement from the trail.
2. Replicat separates comments, if any, from the main statement.
3. Replicat searches for `DDLOPTIONS REMOVECOMMENTS BEFORE`. If it is present, Replicat removes the comments from the DDL statement.
4. Replicat evaluates the DDL synchronization scope to determine if the DDL qualifies for name mapping. Anything else is of `OTHER` scope.
5. Replicat evaluates the `MAP` statements in the parameter file. If the source base object name for this DDL (as read from the trail) appears in any of the `MAP` statements, the operation is marked as `MAPPED` in scope. Otherwise it is marked as `UNMAPPED` in scope.
6. Replicat replaces the source base object name with the base object name that is specified in the `TARGET` clause of the `MAP` statement.
7. If there is a derived object, Replicat searches for `DDLOPTIONS MAPDERIVED`. If it is present, Replicat replaces the source derived name with the target derived name from the `MAP` statement.
8. Replicat checks the DDL parameter for `INCLUDE` and `EXCLUDE` clauses, and it evaluates the DDL parameter criteria contained in them. All options must evaluate to `TRUE` in order for the `INCLUDE` or `EXCLUDE` to evaluate to `TRUE`. The following occurs:
 - If any `EXCLUDE` clause evaluates to `TRUE`, Replicat discards the DDL statement and starts evaluating another DDL statement. In this case, the processing steps start over.

- If any `INCLUDE` clause evaluates to `TRUE`, or if the `DDL` parameter does not have any `INCLUDE` or `EXCLUDE` clauses, Replicat includes the DDL statement, and the processing logic continues.
9. Replicat searches for the `DDLSUBST` parameter and evaluates the `INCLUDE` and `EXCLUDE` clauses. If the options in those clauses add up to `TRUE`, Replicat performs string substitution. Replicat evaluates the DDL statement against each `DDLSUBST` parameter in the parameter file. For all true `DDLSUBST` specifications, Replicat performs string substitution in the order that the `DDLSUBST` parameters are listed in the file.
 10. Now that `DDLSUBST` has been processed, Replicat searches for the `REMOVECOMMENTS AFTER` parameter. If it is present, Replicat removes the comments from the DDL statement.
 11. Replicat executes the DDL statement on the target database.
 12. If there are no errors, Replicat processes the next DDL statement. If there are errors, Replicat performs the following steps.
 13. Replicat analyzes the `INCLUDE` and `EXCLUDE` rules in the Replicat `DDLERROR` parameters in the order that they appear in the parameter file. If Replicat finds a rule for the error code, it applies the specified error handling; otherwise, it applies `DEFAULT` handling.
 14. If the error handling does not enable the DDL statement to succeed, Replicat does one of the following: abends, ignores the operation, or discards it as specified in the rules.

**Note:**

If there are multiple targets for the same source in a `MAP` statement, the processing logic executes for each one.

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 `ADDTRANDATA` to add supplemental logging was applied.

To view a report, use the `VIEW REPORT` command in GGSCI.

```
VIEW REPORT group
```

- [Viewing DDL Reporting in Replicat](#)
- [Viewing DDL Reporting in Extract](#)
- [Statistics in the Process Reports](#)

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 `GGSCI`.

```
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 Tools that Support Trigger-Based DDL Capture

This section documents the additional tools available to support trigger-based capture.

- [Tracing the DDL Trigger](#)
- [Viewing Metadata in the DDL History Table](#)
- [Handling DDL Trigger Errors](#)

Tracing the DDL Trigger

To trace the activity of the Oracle GoldenGate DDL trigger, use the following tools.

- `ggs_ddl_trace.log` trace file: Oracle GoldenGate creates a trace file in the `USER_DUMP_DEST` directory of Oracle. On RAC, each node has its own trace file that captures DDL tracing for that node. You can query the trace file as follows:

```
select value from sys.v_$parameter where name = 'user_dump_dest';
```
- `ddl_tracelevel` script: Edit and run this script to set the trace level. A value of `None` generates no DDL tracing, except for fatal errors and installation logging. The default value of `0` generates minimal tracing information. A value of `1` or `2` generates a much larger amount of information in the trace file. Do not use `1` or `2` unless requested to do so by a Oracle GoldenGate Technical Support analyst as part of a support case.
- `ddl_cleartrace` script: Run this script on a regular schedule to prevent the trace file from consuming excessive disk space as it expands. It deletes the file, but Oracle GoldenGate will create another one. The DDL trigger stops writing to the trace file when the Oracle directory gets low on space, and then resumes writing when space is available again. This script is in the Oracle GoldenGate directory. Back up the trace file before running the script.

Viewing Metadata in the DDL History Table

Use the `DUMPDDL` command in GGSCI to view the information that is contained in the DDL history table. This information is stored in proprietary format, but you can export it in human-readable form to the screen or to a series of SQL tables that you can query. The information in the DDL history table is the same as that used by the Extract process.

Handling DDL Trigger Errors

Use the `params.sql` non-executable script to handle failures of the Oracle GoldenGate DDL trigger in relation to whether the source DDL fails or succeeds. The `params.sql` script is in the root Oracle GoldenGate directory. The parameters to use are the following:

- **ddl_fire_error_in_trigger:** If set to `TRUE`, failures of the Oracle GoldenGate DDL trigger are raised with a Oracle GoldenGate error message and a database error message to the source end-user application. The source operations fails.

If set to `FALSE`, no errors are raised, and a message is written to the trigger trace file in the Oracle GoldenGate directory. The source operation succeeds, but no DDL is replicated. The target application will eventually fail if subsequent data changes do not match the old target object structure. The default is `FALSE`.
- **ddl_cause_error:** If set to `TRUE`, tests the error response of the trigger by deliberately causing an error. To generate the error, Oracle GoldenGate attempts to `SELECT` zero rows without exception handling. Revert this flag to the default of `FALSE` after testing is done.

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.



Note:

EBR support is limited to Integrated Dictionary; it is not supported when using a DDL trigger.

11

Creating Process Groups

This chapter contains instructions for creating Oracle GoldenGate process groups, collectively known as the "change-synchronization" processes. At minimum, you will create one primary Extract, one data pump, and one Replicat process group.

Topics:

- [Prerequisites](#)
This chapter assumes you have installed Oracle GoldenGate, understand the different processing options available to you, and have performed the following prerequisite configuration steps before proceeding to configure Oracle GoldenGate process groups.
- [Registering Extract with the Mining Database](#)
If you are using Extract in integrated mode, you need to create a database logmining server to capture redo data. You do this from the GGSCI interface by registering the primary Extract process with the mining database.
- [Add the Primary Extract](#)
The primary Extract writes to a trail.
- [Add the Local Trail](#)
These steps add the local trail to which the primary Extract writes captured data.
- [Add the Data Pump Extract Group](#)
These steps add the data pump that reads the local trail and sends the data to the target.
- [Add the Remote Trail](#)
Although it is read by Replicat, this trail must be associated with the data pump, so it must be added on the source system, not the target.
- [Add the Replicat Group](#)
These steps add the Replicat group that reads the remote trail and applies the data changes to the target Oracle Database.

Prerequisites

This chapter assumes you have installed Oracle GoldenGate, understand the different processing options available to you, and have performed the following prerequisite configuration steps before proceeding to configure Oracle GoldenGate process groups.

- [Establishing Oracle GoldenGate Credentials](#)
- [Preparing the Database for Oracle GoldenGate](#)
- [Configuring Capture in Integrated Mode](#)
- [Configuring Capture in Classic Mode](#)
- [Configuring Oracle GoldenGate Apply](#)
- [Configuring DDL Support](#) (to use DDL support)

Registering Extract with the Mining Database

If you are using Extract in integrated mode, you need to create a database logmining server to capture redo data. You do this from the GGSCI interface by registering the primary Extract process with the mining database.

The creation of the logmining server extracts a snapshot of the source database in the redo stream of the source database. In a source multitenant container database, you register Extract with each of the pluggable databases that you want to include for capture.

⚠ WARNING:

Make certain that you know the earliest SCN of the log stream at which you want Extract to begin processing. Extract cannot have a starting SCN value that is lower than the first SCN that is specified when the underlying database capture process is created with the `REGISTER EXTRACT` command. You can use the `SCN` option

1. Log into the mining database then use the commands appropriate to your environment. The use of `DBLOGIN` always refers to the source database.

Command for source database deployment:

```
DBLOGIN USERIDALIAS alias
```

Command for downstream mining database deployment:

```
DBLOGIN USERIDALIAS alias  
MININGDBLOGIN USERIDALIAS alias2
```

Where: *alias* specifies the alias of the database login credential that is assigned to Extract. This credential must exist in the Oracle GoldenGate credential store. For more information, see [Establishing Oracle GoldenGate Credentials](#). For more information about `DBLOGIN` and `MININGDBLOGIN`, see *Reference for Oracle GoldenGate*.

2. Register the Extract process with the mining database.

```
REGISTER EXTRACT group DATABASE [CONTAINER (container[, ...])] [SCN  
system_change_number]
```

Where:

- *group* is the name of the Extract group.
- `CONTAINER (container[, ...])` specifies a pluggable database (PDB) within a multitenant container database, or a list of PDBs separated with commas. The specified PDBs must exist before the `REGISTER` command is executed. Extract will capture only from the PDBs that are listed in this command. For example, the following command registers PDBs `mypdb1` and `mypdb4`. Changes from any other PDBs in the multitenant container database are ignored by Oracle GoldenGate.


```
REGISTER EXTRACT myextract DATABASE CONTAINER (mypdb1, mypdb4, mypdb5)
```

You can add or drop pluggable databases at a later date by stopping Extract, issuing a `DBLOGIN` command, and then issuing `REGISTER EXTRACT` with the `{ADD | DROP}` `CONTAINER` option of `DATABASE`.

 **Note:**

Adding `CONTAINERS` at particular SCN on an existing Extract is not supported.

- Registers Extract to begin capture at a specific SCN in the past. Without this option, capture begins from the time that `REGISTER EXTRACT` is issued. The specified SCN must correspond to the begin SCN of a dictionary build operation in a log file. You can issue the following query to find all valid SCN values:

```
SELECT first_change#
       FROM v$aarchived_log
       WHERE dictionary_begin = 'YES' AND
             standby_dest = 'NO' AND
             name IS NOT NULL AND
             status = 'A';
```

- To register additional Extracts with a downstream database for the same source database, issue this `REGISTER` command.

If you want to have more than one extract per source database, you can do that using the `SHARE` with `REGISTER EXTRACT` for better performance and metadata management. The specified SCN must correspond to the SCN where mining should begin in the archive logs.

```
REGISTER EXTRACT group DATABASE [CONTAINER (container[, ...])]
[SCN system_change_number] SHARE
```

 **Note:**

The register process may take a few to several minutes to complete, even though the `REGISTER` command returns immediately.

Add the Primary Extract

The primary Extract writes to a trail.

These steps add the primary Extract that captures change data.

- If using downstream capture, set the RMAN archive log deletion policy to the following value in the source database:

```
CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON ALL STANDBY
```

This must be done before you add the primary Extract.

- Run `GGSCI`.
- If using integrated capture, issue the `DBLOGIN` command.

```
DBLOGIN USERIDALIAS alias
```

Where: *alias* specifies the alias of the database login credential that is assigned to Extract. This credential must exist in the Oracle GoldenGate credential store.

4. Issue the ADD EXTRACT command to add the primary Extract group.

```
ADD EXTRACT group name
{, TRANLOG | , INTEGRATED TRANLOG}
{, BEGIN {NOW | yyyy-mm-dd[ hh:mi:[ss[.cccccc]]]} | SCN value}
[, THREADS n]
```

Where:

- *group name* is the name of the Extract group.
- TRANLOG specifies the transaction log as the data source; *for classic capture only*. See [Example 11-1](#).
- INTEGRATED TRANLOG specifies that Extract receives logical change records through a database logmining server; *for integrated capture only*. See [Example 11-2](#). Before issuing ADD EXTRACT with this option, make certain you logged in to the database with the DBLOGIN command and that you registered this Extract with the database. See [Registering Extract with the Mining Database](#) for more information.
- BEGIN specifies to begin capturing data as of a specific *time*:
 - NOW starts at the first record that is time stamped at the same time that ADD EXTRACT is issued.
 - *yyyy-mm-dd[hh:mi:[ss[.cccccc]]]* starts at an explicit timestamp. Logs from this timestamp must be available. For Extract in integrated mode, the timestamp value must be greater than the timestamp at which the Extract was registered with the database.
 - *SCN value* starts Extract at the transaction in the redo log that has the specified Oracle system change number (SCN). For Extract in integrated mode, the SCN value must be greater than the SCN at which the Extract was registered with the database. See [Registering Extract with the Mining Database](#) for more information.
- THREADS *n* is required in classic capture mode for Oracle Real Application Cluster (RAC), to specify the number of redo log threads being used by the cluster. Extract reads and coordinates each thread to maintain transactional consistency. Not required for integrated capture.

Note:

Additional options are available. See *Reference for Oracle GoldenGate*.

Example 11-1 Classic capture with timestamp start point

```
ADD EXTRACT finance, TRANLOG, BEGIN 2011-01-01 12:00:00.000000
```

Example 11-2 Integrated capture with timestamp start point

```
DBLOGIN USERIDALIAS myalias
ADD EXTRACT finance, INTEGRATED TRANLOG, BEGIN NOW
```

Add the Local Trail

These steps add the local trail to which the primary Extract writes captured data.

In GGSCI on the source system, issue the `ADD EXTTRAIL` command:

```
ADD EXTTRAIL pathname, EXTRACT group name
```

Where:

- `EXTTRAIL` specifies that the trail is to be created on the local system.
- `pathname` is the relative or fully qualified name of the trail, including the two-character name.
- `EXTRACT group name` is the name of the primary Extract group.

**Note:**

Oracle GoldenGate creates this trail automatically during processing.

Example 11-3

```
ADD EXTTRAIL /ggs/dirdat/lt, EXTRACT finance
```

Add the Data Pump Extract Group

These steps add the data pump that reads the local trail and sends the data to the target.

In GGSCI on the source system, issue the `ADD EXTRACT` command.

```
ADD EXTRACT group name, EXTTRAILSOURCE trail name
```

Where:

- `group name` is the name of the Extract group.
- `EXTTRAILSOURCE trail name` is the relative or fully qualified name of the local trail.

Example 11-4

```
ADD EXTRACT financep, EXTTRAILSOURCE c:\ggs\dirdat\lt
```

Add the Remote Trail

Although it is read by Replicat, this trail must be associated with the data pump, so it must be added on the source system, not the target.

These steps add the remote trail:

In GGSCI on the source system, issue the following command:

```
ADD RMTTRAIL pathname, EXTRACT group name
```

Where:

- RMTTRAIL specifies that the trail is to be created on the target system.
- *pathname* is the relative or fully qualified name of the trail, including the two-character name.
- EXTRACT *group name* is the name of the data-pump Extract group.

**Note:**

Oracle GoldenGate creates this trail automatically during processing.

Example 11-5

```
ADD RMTTRAIL /ggs/dirdat/rt, EXTRACT financep
```

Add the Replicat Group

These steps add the Replicat group that reads the remote trail and applies the data changes to the target Oracle Database.

1. Run GGSCI on the target system.
2. If using integrated Replicat, issue the DBLOGIN command to log into the database from GGSCI.

```
DBLOGIN USERIDALIAS alias
```

Where: *alias* specifies the alias of the database login credential that is assigned to Replicat. This credential must exist in the Oracle GoldenGate credential store. For more information, see [Establishing Oracle GoldenGate Credentials](#)

3. Issue the ADD REPLICAT command with the following syntax.

```
ADD REPLICAT group name, [INTEGRATED,] EXTTRAIL pathname
```

Where:

- *group name* is the name of the Replicat group.
- INTEGRATED creates an integrated Replicat group.
- EXTTRAIL *pathname* is the relative or fully qualified name of the remote trail, including the two-character name.

For more information, see *Reference for Oracle GoldenGate*.

Example 11-6 Adds a Nonintegrated Replicat

```
ADD REPLICAT financer, EXTTRAIL c:\ggs\dirdat\rt
```

Example 11-7 Adds an Integrated Replicat

```
ADD REPLICAT financer, INTEGRATED, EXTTRAIL c:\ggs\dirdat\rt
```

Instantiating Oracle GoldenGate Replication

This chapter contains instructions for configuring and performing an instantiation of the replication environment to establish and maintain a synchronized state between two or more databases. In a synchronized state, the source and target objects contain identical or appropriately corresponding values, depending on whether any conversion or transformation is performed on the data before applying it to the target objects.

Topics:

- [Overview of the Instantiation Process](#)
In the instantiation procedure, you make a copy of the source data and load the copy to the target database.
- [Prerequisites for Instantiation](#)
The following steps must be taken before starting any Oracle GoldenGate processes or native database load processes.
- [Configuring the Initial Load](#)
Oracle GoldenGate supports these load methods in this section specifically for Oracle Database.
- [Performing the Target Instantiation](#)
This procedure instantiates the target tables while Oracle GoldenGate captures ongoing transactional changes on the source and stores them until they can be applied on the target.
- [Monitoring and Controlling Processing After the Instantiation](#)
After the target is instantiated and replication is in effect, you can control processes and view the overall health of the replication environment.
- [Verifying Synchronization](#)
To verify that the source and target data are synchronized, you can use the Oracle GoldenGate Veridata product or use your own scripts to select and compare source and target data.
- [Backing up the Oracle GoldenGate Environment](#)
After you start Oracle GoldenGate processing, an effective backup routine is critical to preserving the state of processing in the event of a failure. Unless the Oracle GoldenGate working files can be restored, the entire replication environment must be re-instantiated, complete with new initial loads.

Overview of the Instantiation Process

In the instantiation procedure, you make a copy of the source data and load the copy to the target database.

The initial load captures a point-in-time snapshot of the data, while Oracle GoldenGate maintains that consistency by applying transactional changes that occur while the static data is being loaded. After instantiation is complete, Oracle GoldenGate maintains the synchronized state throughout ongoing transactional changes.

When you instantiate Oracle GoldenGate processing, it is recommended that you do so first in a test environment before deploying live on your production machines. This is especially

important in an active-active or high availability configuration, where trusted source data may be touched by the replication processes. Testing enables you to find and resolve any configuration mistakes or data issues without the need to interrupt user activity for re-loads on the target or other troubleshooting activities. Testing also ensures that your instantiation process is configured properly. Parameter files can be copied to the production equipment after successful testing, and then you can perform a predictable instantiation with production data.

Prerequisites for Instantiation

The following steps must be taken before starting any Oracle GoldenGate processes or native database load processes.

- [Configuring and Adding Change Synchronization Groups](#)
- [Disabling DDL Processing](#)
- [Adding Collision Handling](#)
- [Preparing the Target Tables](#)

Configuring and Adding Change Synchronization Groups

To perform an instantiation of the target database and the replication environment, the online change capture and apply groups must exist and be properly configured. See:

- [Configuring Capture in Integrated Mode](#)
- [Configuring Capture in Classic Mode](#)
- [Configuring Oracle GoldenGate Apply](#)
- [Creating Process Groups](#)

Disabling DDL Processing

You must disable DDL activities before performing an instantiation. You can resume DDL after the instantiation is finished. See [Disabling DDL Processing Temporarily](#) for instructions.

Adding Collision Handling

This prerequisite applies to the following instantiation methods:

- [Configuring a Direct Bulk Load to SQL*Loader](#)
- [Configuring a Load from an Input File to SQL*Loader](#)

This prerequisite *does not* apply to the instantiation method described in [Configuring a Load with an Oracle Data Pump](#).

If the source database will remain active during one of those initial load methods, collision-handling logic must be added to the Replicat parameter file. This logic handles conflicts that occur because static data is being loaded to the target tables while Oracle GoldenGate replicates transactional changes to those tables.

To handle collisions, add the `HANDLECOLLISIONS` parameter to the Replicat parameter file to resolve these collisions:

- `INSERT` operations for which the row already exists
- `UPDATE` and `DELETE` operations for which the row does not exist

`HANDLECOLLISIONS` should be removed from the Replicat parameter file at the end of the instantiation steps (as prompted in the instructions).

To use the `HANDLECOLLISIONS` function to reconcile incremental data changes with the load, each target table must have a primary or unique key. If you cannot create a key through your application, use the `KEYCOLS` option of the `TABLE` and `MAP` parameters to specify columns as a substitute key for Oracle GoldenGate to use. If you cannot create keys, the affected source table must be quiesced for the load.

Preparing the Target Tables

The following are suggestions that can make the load go faster and help you to avoid errors.

- **Data:** Make certain that the target tables are empty. Otherwise, there may be duplicate-row errors or conflicts between existing rows and rows that are being loaded.
- **Indexes:** Remove indexes from the target tables. Indexes are not necessary for the inserts performed by the initial load process and will slow it down. You can add back the indexes after the load is finished.

Configuring the Initial Load

Oracle GoldenGate supports these load methods in this section specifically for Oracle Database.

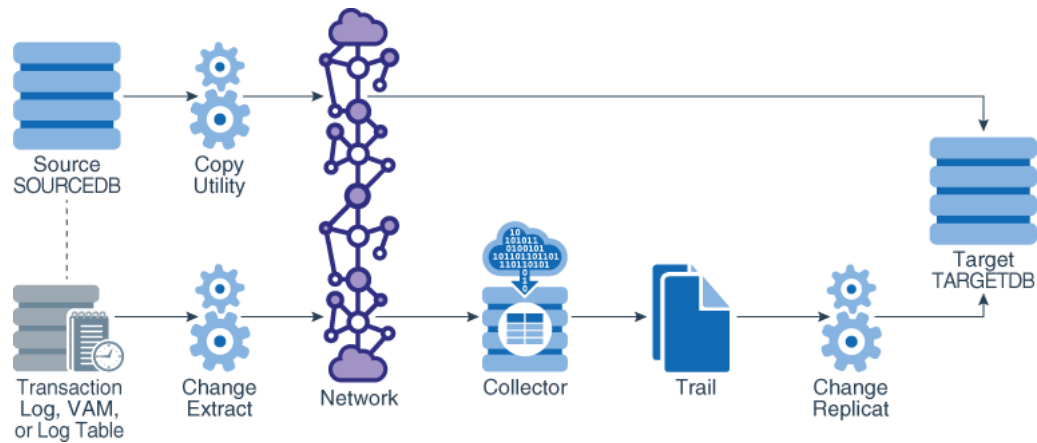
Select a method and follow its configuration steps to create the load processes and parameter files. To work with parameter files, see *Using Oracle GoldenGate Parameter Files in Administering Oracle GoldenGate*.

- [Configuring a Load with an Oracle Data Pump](#)
- [Configuring a Direct Bulk Load to SQL*Loader](#)
- [Configuring a Load from an Input File to SQL*Loader](#)

Configuring a Load with an Oracle Data Pump

This method uses the Oracle Data Pump utility to establish the target data. You start Extract, the data pumps, and Replicat at the SCN at which the copy stopped. Transactions that were included in the copy are skipped to avoid collisions from integrity violations. From the process start point, Oracle GoldenGate maintains data synchronization.

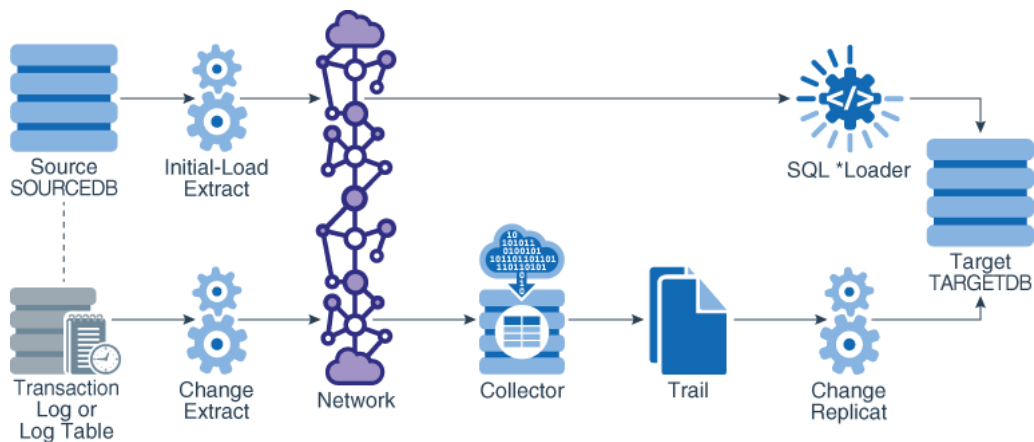
No initial-load Oracle GoldenGate processes are required for this method.



Configuring a Direct Bulk Load to SQL*Loader

With this method, you configure and run an Oracle GoldenGate initial-load Extract to extract complete source records and send them directly to an initial-load Replicat task. The initial-load Replicat task communicates with SQL*Loader to load data as a direct-path bulk load. Data mapping and transformation can be done by either the initial-load Extract or initial-load Replicat, or both. During the load, the change-synchronization groups that you configured in [Configuring Capture in Integrated Mode](#) or [Configuring Capture in Classic Mode](#) and [Configuring Oracle GoldenGate Apply](#) replicate incremental changes, which are then reconciled with the results of the load.

The following diagram shows configuring a direct bulk load to SQL*Loader.



Limitations:

- This method does not support extraction of LOB or LONG data. As an alternative, see [Performing Instantiation From an Input File to SQL*Loader](#).
- This method does not support materialized views that contain LOBs, regardless of their size. It also does not support data encryption.

To Configure a Direct Bulk Load to SQL*Loader

1. Grant `LOCK ANY TABLE` to the Replicat database user on the target Oracle Database.
2. On the source and target systems, run GGSCI.
3. Start Manager on both systems.

```
START MANAGER
```

4. On the source system, create the initial-load Extract.

```
ADD EXTRACT initial-load_Extract, SOURCEISTABLE
```

Where:

- *initial-load_Extract* is the name of the initial-load Extract, up to eight characters.
 - `SOURCEISTABLE` directs Extract to read complete records directly from the source tables.
5. On the source system, create the initial-load Extract parameter file.

```
EDIT PARAMS initial-load_Extract
```

6. Enter the initial-load Extract parameters in the order shown, starting a new line for each parameter statement. This example shows a three-part table name associated with a multitenant container database.

```
EXTRACT initext
USERIDALIAS tiger1
RMTHOST fin1, MGRPORT 7809 ENCRYPT AES192, KEYNAME securekey2
RMTTASK replicat, GROUP initrep
TABLE hq.hr.*;
```

Parameter	Description
EXTRACT <i>initial-load_Extract</i>	Specifies the name of the initial-load Extract, as stated with <code>ADD EXTRACT</code> .
USERIDALIAS <i>alias</i>	Specifies the alias of the database login credential that is assigned to Extract. This credential must exist in the Oracle GoldenGate credential store. For more information, see Establishing Oracle GoldenGate Credentials
RMTHOST <i>hostname</i> , MGRPORT <i>portnumber</i> [, ENCRYPT <i>algorithm</i> KEYNAME <i>keyname</i>]	Specifies the target system, the port where Manager is running, and optional encryption of data across TCP/IP.
RMTTASK REPLICAT, GROUP <i>initial-load_Replicat</i>	Specifies the process type (must be <code>REPLICAT</code>) and the name of the initial-load Replicat. Directs Manager on the target system to dynamically start the initial-load Replicat as a one-time task.
TABLE <i>[container.]schema.table;</i>	Specifies the tables to capture. <ul style="list-style-type: none"> • <i>container</i> is the name of the pluggable database, if this is a multitenant container database. You can use the <code>SOURCECATALOG</code> parameter to specify a default pluggable database instead of using three-part names. • <i>schema</i> is the schema name. • <i>table</i> is the table name.

7. Save and close the file.

8. On the target system, create the initial-load Replicat.

```
ADD REPLICAT initial-load Replicat, SPECIALRUN
```

Where:

- *initial-load Replicat* is the name of the initial-load Replicat task.
- SPECIALRUN identifies the initial-load Replicat as a one-time task, not a continuous process.

9. On the target system, create the initial-load Replicat parameter file.

```
EDIT PARAMS initial-load Replicat
```

10. Enter the initial-load Replicat parameters in the order shown, starting a new line for each parameter statement. This example shows a three-part source table name associated with a multitenant container database.

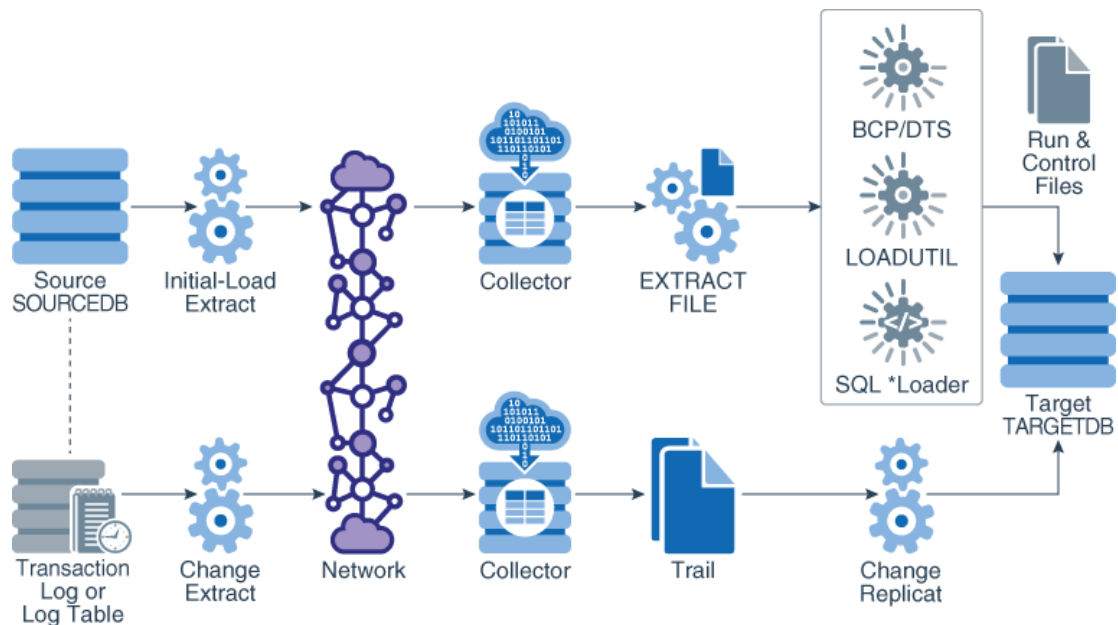
```
REPLICAT initrep
USERIDALIAS tiger2
BULKLOAD
ASSUMETARGETDEFS
MAP hq.hr.*, TARGET hr2.*;
```

Parameter	Description
REPLICAT <i>initial-load Replicat</i>	Specifies the name of the initial-load Replicat task, as stated with ADD REPLICAT.
USERIDALIAS <i>alias</i>	Specifies the alias of the database login credential that is assigned to Replicat. This credential must exist in the Oracle GoldenGate credential store.
BULKLOAD	Directs Replicat to interface directly with the Oracle SQL*Loader interface.
ASSUMETARGETDEFS	Assumes the source and target tables are identical, including semantics. If source and target definitions are different, you must create and specify a source-definitions file that both the change-synchronization and initial-load processes will use.
MAP <i>[container.]schema.table</i> , TARGET <i>schema.table</i> ;	Specifies a relationship between a source and target table or tables. <ul style="list-style-type: none"> • If the source is a multitenant container database, <i>container</i> is the name of the pluggable database that contains the source objects specified with this MAP statement. You can use the SOURCECATALOG parameter to specify a default source pluggable database instead of using three-part names. • <i>schema</i> is the schema name. • <i>table</i> is the table name.

Configuring a Load from an Input File to SQL*Loader

With this method, an initial-load Extract extracts source records from the source tables and writes them to an extract file in external ASCII format. The files are read by SQL*Loader. During the load, the change-synchronization groups that you configured in Chapter 4 replicate incremental changes, which are then reconciled with the results of the load. As part of the load procedure, Oracle GoldenGate uses the initial-load Replicat to create run and control files required by the database utility. Any data

transformation must be performed by the initial-load Extract on the source system because the control files are generated dynamically and cannot be pre-configured with transformation rules.



To Configure a Load from File to SQL*Loader

1. On the source and target systems, run GGSCI.
2. Start Manager on both systems.

```
START MANAGER
```

3. On the source system, create the initial-load Extract parameter file.

```
EDIT PARAMS initial-load Extract
```

4. Enter the initial-load Extract parameters in the order shown, starting a new line for each parameter statement. This example shows a three-part table name associated with a multitenant container database.

```
SOURCEISTABLE
USERIDALIAS tiger1
RMTHOST fin1, MGRPORT 7809 ENCRYPT AES192, KEYNAME securekey2
ENCRYPTTRAIL AES192
FORMATASCII, SQLLOADER
RMTFILE /ggs/dirdat/ie
TABLE hq.hr.*;
```

Parameter	Description
SOURCEISTABLE	Designates Extract as an initial load process that extracts records directly from the source tables.
USERIDALIAS <i>alias</i>	Specifies the alias of the database login credential that is assigned to Extract. This credential must exist in the Oracle GoldenGate credential store, see Establishing Oracle GoldenGate Credentials

Parameter	Description
RMTHOST <i>hostname</i> , MGRPRT <i>portnumber</i> [, ENCRYPT <i>algorithm</i> KEYNAME <i>keyname</i>]	Specifies the target system, the port where Manager is running, and optional encryption of data across TCP/IP.
ENCRYPTTRAIL <i>algorithm</i>	Encrypts the data in the remote file. For more information.
FORMATASCII, SQLLOADER	Produces a fixed-length, ASCII-formatted remote file that is compatible with SQL*Loader. This parameter must be listed before RMTFILE.
RMTFILE <i>path</i>	Specifies the absolute or full path name of an extract file that Extract creates and to which it writes the load data.
TABLE <i>[container.]schema.table;</i>	Specifies the tables to capture. <ul style="list-style-type: none"> • <i>container</i> is the name of the pluggable database, if this is a multitenant container database. You can use the SOURCECATALOG parameter to specify a default pluggable database instead of using three-part names. • <i>schema</i> is the schema name. • <i>table</i> is the table name.

5. Save and close the parameter file.
6. On the target system, create the initial-load Replicat parameter file.

```
EDIT PARAMS initial-load Replicat
```

7. Enter the initial-load Replicat parameters in the order shown, starting a new line for each parameter statement. This example shows a three-part source table name associated with a multitenant container database.

```
GENLOADFILES sqlldr.tpl  
USERIDALIAS tiger2  
EXTFILE /ggs/dirdat/ie  
ASSUMETARGETDEFS  
MAP hq.hr.*, TARGET hr2.*;
```

Parameter	Description
GENLOADFILES <i>template</i>	Generates run and control files for the database utility.
USERIDALIAS <i>alias</i>	Specifies the alias of the database login credential of the user that is assigned to Replicat. This credential must exist in the Oracle GoldenGate credential store, see Establishing Oracle GoldenGate Credentials
EXTFILE <i>path</i>	Specifies the extract file that you specified with the Extract parameter RMTFILE.
ASSUMETARGETDEFS	Assumes the source and target tables are identical, including semantics. If source and target definitions are different, you must create and specify a source-definitions file that both the change-synchronization and initial-load processes will use.

Parameter	Description
MAP <i>[container.]schema.table</i> , TARGET <i>schema.table</i> ;	<p>Specifies a relationship between a source and target table or tables.</p> <ul style="list-style-type: none"> If the source is a multitenant container database, <i>container</i> is the name of the pluggable database that contains the source objects specified with this MAP statement. You can use the SOURCECATALOG parameter to specify a default source pluggable database instead of using three-part names. <i>schema</i> is the schema name. <i>table</i> is the table name.

8. Save and close the parameter file.

Performing the Target Instantiation

This procedure instantiates the target tables while Oracle GoldenGate captures ongoing transactional changes on the source and stores them until they can be applied on the target.

By the time you perform the instantiation of the target tables, the entire Oracle GoldenGate environment should be configured for change capture and delivery, as should the initial-load processes if using Oracle GoldenGate as an initial-load utility.



Note:

The first time that Extract starts in a new Oracle GoldenGate configuration, any open source transactions will be skipped. Only transactions that begin after Extract starts are captured.

- [Performing Instantiation with Oracle Data Pump](#)
- [Performing Instantiation with Direct Bulk Load to SQL*Loader](#)
- [Performing Instantiation From an Input File to SQL*Loader](#)

Performing Instantiation with Oracle Data Pump

To perform instantiation with Oracle Data Pump, see My Oracle Support document 1276058.1. To obtain this document, do the following:

1. Go to <http://support.oracle.com>.
2. Under Sign In, select your language and then log in with your Oracle Single Sign-On (SSO).
3. On the Dashboard, expand the Knowledge Base heading.
4. Under Enter Search Terms, paste or type the document ID of 1276058.1 and then click **Search**.
5. In the search results, select **Oracle GoldenGate Best Practices: Instantiation from an Oracle Source Database [Article ID 1276058.1]**.
6. Click the link under Attachments to open the article.

Performing Instantiation with Direct Bulk Load to SQL*Loader

1. On the source system, run GGSCI.
2. Start the primary change-capture Extract group.


```
START EXTRACT group
```
3. Start the data-pump Extract group.

```
START EXTRACT data_pump
```
4. If replicating sequence values:
 - Issue the `DBLOGIN` command with the alias of a user in the credential store who has `EXECUTE privilege on update.Sequence`.

```
DBLOGIN USERIDALIAS alias
```
 - Issue the following command to update each source sequence and generate redo. From the redo, Replicat performs initial synchronization of the sequences on the target.

```
FLUSH SEQUENCE [container.]schema.sequence
```
5. Start the initial-load Extract.

```
START EXTRACT initial-load_Extract
```

 **WARNING:**

Do not start the initial-load Replicat. The Manager process starts it automatically and terminates it when the load is finished.
6. On the target system, run GGSCI.
7. Issue the `VIEW REPORT` command to determine when the initial load to SQL*Loader is finished.

```
VIEW REPORT initial-load_Extract
```
8. When the load is finished, start the change-data Replicat group.

```
START REPLICAT group
```
9. Issue the `INFO REPLICAT` command, and continue to issue it until it shows that Replicat posted all of the change data that was generated during the initial load. For example, if the initial-load Extract stopped at 12:05, make sure Replicat posted data up to that time.

```
INFO REPLICAT group
```
10. Turn off `HANDLECOLLISIONS` for the change-delivery Replicat to disable initial-load error handling.

```
SEND REPLICAT group, NOHANDLECOLLISIONS
```
11. Edit the change-delivery Replicat parameter file to remove the `HANDLECOLLISIONS` parameter.

```
EDIT PARAMS group
```

12. Save and close the parameter file.

From this point forward, Oracle GoldenGate continues to synchronize data changes.

Performing Instantiation From an Input File to SQL*Loader

Note:

The SQL*Loader method is not recommended if the data has multibyte characters, especially when the character set of the operating system is different from the database character set.

1. On the source system, run GGSCI.
2. Start the primary change-capture Extract group.

```
START EXTRACT group
```
3. Start the data-pump Extract group.

```
START EXTRACT data_pump
```
4. If replicating sequence values:
 - Issue the `DBLOGIN` command with the alias of a user in the credential store who has `EXECUTE privilege on update.Sequence`.

```
DBLOGIN USERIDALIAS alias
```
 - Issue the following command to update each source sequence and generate redo. From the redo, Replicat performs initial synchronization of the sequences on the target.

```
FLUSH SEQUENCE [container.]schema.sequence
```
5. From the Oracle GoldenGate installation directory on the source system, start the initial-load Extract from the command line of the operating system (not GGSCI).

UNIX and Linux:

```
$ /OGG_directory/extract paramfile dirprm/initial-load_Extract.prm reportfile path
```

Windows:

```
C:\> OGG_directory\extract paramfile dirprm\initial-load_Extract.prm reportfile path
```

Where: `initial-load_Extract` is the name of the initial-load Extract and `path` is the relative or fully qualified path where you want the Extract report file to be created.

6. Wait until the initial extraction from the source is finished. Verify its progress and results by viewing the Extract report file from the command line.
7. On the target system, start the initial-load Replicat.

UNIX and Linux:

```
$ /OGG_directory/replicat paramfile dirprm/initial-load_Replicat name.prm reportfile path
```

Windows:

```
C:\> OGG directory\replicat paramfile dirprm\initial-load_Replicat.prm  
reportfile path
```

Where: `initial-load Extract` is the name of the initial-load Replicat and `path` is the relative or fully qualified path where you want the Replicat report file to be created.

8. When the initial-load Replicat stops, verify its results by viewing the Replicat report file from the command line.
9. Using the ASCII-formatted file and the run and control files that the initial-load Replicat created, load the data with SQL*Loader.
10. When the load is finished, start the change-delivery Replicat group.

```
START REPLICAT group
```

11. Issue the `INFO REPLICAT` command, and continue to issue it until it shows that Replicat posted all of the change data that was generated during the initial load. For example, if the initial-load Extract stopped at 12:05, make sure Replicat posted data up to that time.

```
INFO REPLICAT group
```

12. Turn off `HANDLECOLLISIONS` for the change-delivery Replicat to disable initial-load error handling.

```
SEND REPLICAT group, NOHANDLECOLLISIONS
```

13. Edit the change-delivery Replicat parameter file to remove the `HANDLECOLLISIONS` parameter.

```
EDIT PARAMS group
```

14. Save and close the parameter file.

From this point forward, Oracle GoldenGate continues to synchronize data changes.

Monitoring and Controlling Processing After the Instantiation

After the target is instantiated and replication is in effect, you can control processes and view the overall health of the replication environment.

If you configured Replicat in integrated mode, you can use the `STATS REPLICAT` command to view statistics on the number of transactions that are applied in integrated mode as compared to those that are applied in direct apply mode.

```
STATS REPLICAT group
```

The output of this command shows the number of transactions applied, the number of transactions that were redirected to direct apply, and the direct transaction ratio, among other statistics. The statistics help you determine whether integrated Replicat is performing as intended. If the environment is satisfactory and there is a high ratio of direct apply operations, consider using nonintegrated Replicat. You can configure parallelism with nonintegrated Replicat.

 **Note:**

To ensure realistic statistics, view apply statistics only after you are certain that the Oracle GoldenGate environment is well established, that configuration errors are resolved, and that any anticipated processing errors are being handled properly.

You can also view runtime statistics for integrated Replicat in the `V$`views for each of the inbound server components.

- The reader statistics are recorded in `V$GG_APPLY_READER` and include statistics on number of messages read, memory used, and dependency counts.
- The apply coordinator statistics are recorded in `V$GG_APPLY_COORDINATOR` and record statistics at the transaction level.
- The apply server statistics are recorded in `V$GG_APPLY_SERVER`. This view records information for each of the apply server processes (controlled by `parallelism` and `max_parallelism` parameters) as separate rows. The statistics for each apply server are identified by the `SERVER_ID` column. If a `SERVER_ID` of 0 exists, this represents an aggregate of any apply servers that exited because the workload was reduced.
- Statistics about the number of messages received by the database from Replicat are recorded in the `V$GG_APPLY_RECEIVER` table.

To control processes, see *Controlling Oracle GoldenGate Processes* in *Administering Oracle GoldenGate*.

To ensure that all processes are running properly and that errors are being handled according to your error handling rules, see *Handling Processing Errors* in *Administering Oracle GoldenGate*. Oracle GoldenGate provides commands and logs to view process status, lag, warnings, and other information.

To know more about querying the following views, see Oracle Database Reference.

- `V$GOLDENGATE_TABLE_STATS` to see statistics for DML and collisions that occurred for each replicated table that the inbound server processed.
- `V$GOLDENGATE_TRANSACTION` to see information about transactions that are being processed by Oracle GoldenGate inbound servers.

Verifying Synchronization

To verify that the source and target data are synchronized, you can use the Oracle GoldenGate Veridata product or use your own scripts to select and compare source and target data.

Backing up the Oracle GoldenGate Environment

After you start Oracle GoldenGate processing, an effective backup routine is critical to preserving the state of processing in the event of a failure. Unless the Oracle GoldenGate

working files can be restored, the entire replication environment must be re-instantiated, complete with new initial loads.

As a best practice, include the entire Oracle GoldenGate home installation in your backup routines. There are too many critical sub-directories, as well as files and programs at the root of the directory, to keep track of separately. In any event, the most critical files are those that consume the vast majority of backup space, and therefore it makes sense just to back up the entire installation directory for fast, simple recovery.

13

Managing the DDL Replication Environment

This chapter contains instructions for making changes to the database environment or the Oracle GoldenGate environment when the Oracle GoldenGate DDL trigger is being used to support DDL replication. See [Installing Trigger-Based DDL Capture](#) for more information about the DDL objects.

For instructions on configuring Oracle GoldenGate DDL support, see [Configuring DDL Support](#).



Note:

This chapter is only relevant for classic capture mode or integrated capture mode in which trigger-based DDL capture is being used.

Topics:

- [Disabling DDL Processing Temporarily](#)
You must disable DDL activities before performing an instantiation or other tasks, if directed.
- [Enabling and Disabling the DDL Trigger](#)
You can enable and disable the trigger that captures DDL operations without making any configuration changes within Oracle GoldenGate.
- [Maintaining the DDL Marker Table](#)
You can purge rows from the marker table at any time. It does not keep DDL history.
- [Deleting the DDL Marker Table](#)
Do not delete the DDL marker table unless you want to discontinue synchronizing DDL.
- [Maintaining the DDL History Table](#)
You can purge the DDL history table to control its size, but do so carefully.
- [Deleting the DDL History Table](#)
The history table and the DDL trigger are interdependent. An attempt to drop the history table fails if the DDL trigger is enabled. This is a safety measure to prevent the trigger from becoming invalid and missing DDL operations.
- [Purging the DDL Trace File](#)
To prevent the DDL trace file from consuming excessive disk space, run the `ddl_cleartrace` script on a regular basis.
- [Applying Database Patches and Upgrades when DDL Support is Enabled](#)
Database patches and upgrades usually invalidate the Oracle GoldenGate DDL trigger and other Oracle GoldenGate DDL objects.
- [Apply Oracle GoldenGate Patches and Upgrades when DDL support is Enabled](#)
Use the following steps to apply a patch or upgrade to the DDL objects.
- [Restoring an Existing DDL Environment to a Clean State](#)
Follow these steps to completely remove, and then reinstall, the Oracle GoldenGate DDL objects.

- [Removing the DDL Objects from the System](#)
This procedure removes the DDL environment and removes the history that maintains continuity between source and target DDL operations.

Disabling DDL Processing Temporarily

You must disable DDL activities before performing an instantiation or other tasks, if directed.

You can resume DDL processing after the task is finished.

1. Disable user DDL operations on the source database.
2. If there are previous DDL replication processes that are still active, make certain that the last executed DDL operation was applied to the target before stopping those processes, so that the load data is applied to objects that have the correct metadata.
3. Comment out the `DDL` parameter in the Extract and Replicat parameter files that you configured for the new Oracle GoldenGate environment. Comment out any other parameters that support DDL.
4. Disable the Oracle GoldenGate DDL trigger, if one is in use. See [Enabling and Disabling the DDL Trigger](#).

Enabling and Disabling the DDL Trigger

You can enable and disable the trigger that captures DDL operations without making any configuration changes within Oracle GoldenGate.

The following scripts control the DDL trigger.

- `ddl_disable`: Disables the trigger. No further DDL operations are captured or replicated after you disable the trigger.
- `ddl_enable`: Enables the trigger. When you enable the trigger, Oracle GoldenGate starts capturing current DDL changes, but does not capture DDL that was generated while the trigger was disabled.

Before running these scripts, disable all sessions that ever issued DDL, including those of the Oracle GoldenGate processes, SQL*Plus, business applications, and any other software that uses Oracle. Otherwise the database might generate an ORA-04021 error. Do not use these scripts if you intend to maintain consistent DDL on the source and target systems.

Maintaining the DDL Marker Table

You can purge rows from the marker table at any time. It does not keep DDL history.

To purge the marker table, use the Manager parameter `PURGEMARKERHISTORY`. Manager gets the name of the marker table from one of the following:

1. The name given with the `MARKERTABLE` parameter in the `GLOBALS` file, if specified.
2. The default name of `GGS_MARKER`.

`PURGEMARKERHISTORY` provides options to specify maximum and minimum lengths of time to keep a row, based on the last modification date.

Deleting the DDL Marker Table

Do not delete the DDL marker table unless you want to discontinue synchronizing DDL.

The marker table and the DDL trigger are interdependent. An attempt to drop the marker table fails if the DDL trigger is enabled. This is a safety measure to prevent the trigger from becoming invalid and missing DDL operations. If you remove the marker table, the following error is generated:

```
ORA-04098: trigger 'SYS.GGS_DDL_TRIGGER_BEFORE' is invalid and failed re-validation
```

The proper way to remove an Oracle GoldenGate DDL object depends on your plans for the rest of the DDL environment. To choose the correct procedure, see one of the following:

- [Restoring an Existing DDL Environment to a Clean State](#)
- [Removing the DDL Objects from the System](#)

Maintaining the DDL History Table

You can purge the DDL history table to control its size, but do so carefully.

The DDL history table maintains the integrity of the DDL synchronization environment. Purges to this table cannot be recovered through the Oracle GoldenGate interface.

1. To prevent any possibility of DDL history loss, make regular full backups of the history table.
2. To ensure that purged DDL can be recovered, enable Oracle Flashback for the history table. Set the flashback retention time well past the point where it could be needed. For example, if your full backups are at most one week old, retain two weeks of flashback. Oracle GoldenGate can be positioned backward into the flashback for reprocessing.
3. If possible, purge the DDL history table manually to ensure that essential rows are not purged accidentally. If you require an automated purging mechanism, use the `PURGEDDLHISTORY` parameter in the Manager parameter file. You can specify maximum and minimum lengths of time to keep a row.

Note:

Temporary tables created by Oracle GoldenGate to increase performance might be purged at the same time as the DDL history table, according to the same rules. The names of these tables are derived from the name of the history table, and their purging is reported in the Manager report file. This is normal behavior.

Deleting the DDL History Table

The history table and the DDL trigger are interdependent. An attempt to drop the history table fails if the DDL trigger is enabled. This is a safety measure to prevent the trigger from becoming invalid and missing DDL operations.

Do not delete the DDL history table unless you want to discontinue synchronizing DDL. The history table contains a record of DDL operations that were issued. Once an Extract switches

from using the DDL trigger to not using the trigger, as when source database redo compatibility is advanced to 11.2.0.4 or greater, these objects can be deleted though *not immediately*. It is imperative that all mining of the redo generated before the compatibility change be complete and that this redo not need to be mined again.

If you remove the history table, the following error is generated:

```
ORA-04098: trigger 'SYS.GGS_DDL_TRIGGER_BEFORE' is invalid and failed re-validation
```

The proper way to remove an Oracle GoldenGate DDL object depends on your plans for the rest of the DDL environment. To choose the correct procedure, see one of the following:

- [Restoring an Existing DDL Environment to a Clean State](#)
- [Removing the DDL Objects from the System](#)

Purging the DDL Trace File

To prevent the DDL trace file from consuming excessive disk space, run the `ddl_cleartrace` script on a regular basis.

This script deletes the trace file, but Oracle GoldenGate will create it again.

The default name of the DDL trace file is `ggs_ddl_trace.log`. It is in the `USER_DUMP_DEST` directory of Oracle. The `ddl_cleartrace` script is in the Oracle GoldenGate directory.

Applying Database Patches and Upgrades when DDL Support is Enabled

Database patches and upgrades usually invalidate the Oracle GoldenGate DDL trigger and other Oracle GoldenGate DDL objects.

Before applying a database patch, do the following.

1. Log in to SQL*Plus as a user that has `SYSDBA` privileges.
2. Disable the Oracle GoldenGate DDL trigger by running the `ddl_disable` script in SQL*Plus.
3. Apply the patch.
4. Enable the DDL trigger by running the `ddl_enable` script in SQL*Plus.

Note:

Database upgrades and patches generally operate on Oracle objects. Because Oracle GoldenGate filters out those objects automatically, DDL from those procedures is not replicated when replication starts again.

To avoid recompile errors after the patch or upgrade, which are caused if the trigger is not disabled before the procedure, consider adding calls to `@ddl_disable` and `@ddl_enable` at the appropriate locations within your scripts.

Apply Oracle GoldenGate Patches and Upgrades when DDL support is Enabled

Use the following steps to apply a patch or upgrade to the DDL objects.

This section explains how to apply Oracle GoldenGate patches and upgrades when DDL support is enabled.

Note:

If the release notes or upgrade documentation for your Oracle GoldenGate release contain instructions similar to those provided in this section, follow those instructions instead of the ones in this section. Do not use this procedure for an upgrade from an Oracle GoldenGate version that does not support DDL statements that are larger than 30K (pre-version 10.4). To upgrade in that case, follow the instructions in [Restoring an Existing DDL Environment to a Clean State](#).

This procedure may or may not preserve the current DDL synchronization configuration, depending on whether the new build requires a clean installation.

1. Run GGSCI. Keep the session open for the duration of this procedure.
2. Stop Extract to stop DDL capture.

```
STOP EXTRACT group
```
3. Stop Replicat to stop DDL replication.

```
STOP REPLICAT group
```
4. Download or extract the patch or upgrade files according to the instructions provided by Oracle GoldenGate.
5. Change directories to the Oracle GoldenGate installation directory.
6. Log in to SQL*Plus as a user that has SYSDBA privileges.
7. Disconnect all sessions that ever issued DDL, including those of Oracle GoldenGate processes, SQL*Plus, business applications, and any other software that uses Oracle. Otherwise the database might generate an ORA-04021 error.
8. Run the `ddl_disable` script to disable the DDL trigger.
9. Run the `ddl_setup` script. You are prompted for the name of the Oracle GoldenGate DDL schema. If you changed the schema name, use the new one.
10. Run the `ddl_enable.sql` script to enable the DDL trigger.
11. In GGSCI, start Extract to resume DDL capture.

```
START EXTRACT group
```
12. Start Replicat to start DDL replication.

```
START REPLICAT group
```

Restoring an Existing DDL Environment to a Clean State

Follow these steps to completely remove, and then reinstall, the Oracle GoldenGate DDL objects.

This procedure creates a new DDL environment and removes any current DDL history.

Note:

Due to object interdependencies, all objects must be removed and reinstalled in this procedure.

1. If you are performing this procedure in conjunction with the installation of a new Oracle GoldenGate version, download and install the Oracle GoldenGate files, and create or update process groups and parameter files as necessary.
2. (Optional) To preserve the continuity of source and target structures, stop DDL activities and then make certain that Replicat finished processing all of the DDL and DML data in the trail. To determine when Replicat is finished, issue the following command until you see a message that there is no more data to process.

```
INFO REPLICAT group
```

Note:

Instead of using `INFO REPLICAT`, you can use the `EVENTACTIONS` option of `TABLE` and `MAP` to stop the Extract and Replicat processes after the DDL and DML has been processed.

3. Run GGSCI.
4. Stop Extract to stop DDL capture.

```
STOP EXTRACT group
```
5. Stop Replicat to stop DDL replication.

```
STOP REPLICAT group
```
6. Change directories to the Oracle GoldenGate installation directory.
7. Log in to SQL*Plus as a user that has SYSDBA privileges.
8. Disconnect all sessions that ever issued DDL, including those of Oracle GoldenGate processes, SQL*Plus, business applications, and any other software that uses Oracle. Otherwise the database might generate an ORA-04021 error.
9. Run the `ddl_disable` script to disable the DDL trigger.
10. Run the `ddl_remove` script to remove the Oracle GoldenGate DDL trigger, the DDL history and marker tables, and other associated objects. This script produces a `ddl_remove_spool.txt` file that logs the script output and a `ddl_remove_set.txt` file that logs environment settings in case they are needed for debugging.

11. Run the `marker_remove` script to remove the Oracle GoldenGate marker support system. This script produces a `marker_remove_spool.txt` file that logs the script output and a `marker_remove_set.txt` file that logs environment settings in case they are needed for debugging.
12. If you are changing the DDL schema for this installation, grant the following permission to the Oracle GoldenGate schema.

```
GRANT EXECUTE ON utl_file TO schema;
```

13. If you are changing the DDL schema for this installation, the schema's default tablespace must be dedicated to that schema; do not allow any other schema to share it. `AUTOEXTEND` must be set to `ON` for this tablespace, and the tablespace must be sized to accommodate the growth of the `GG$DDL_HIST` and `GG$MARKER` tables. The `GG$DDL_HIST` table, in particular, will grow in proportion to overall DDL activity.

 **Note:**

If the DDL tablespace fills up, Extract stops capturing DDL. To cause user DDL activity to fail when that happens, edit the `params.sql` script and set the `ddl_fire_error_in_trigger` parameter to `TRUE`. Stopping user DDL gives you time to extend the tablespace size and prevent the loss of DDL capture. Managing tablespace sizing this way, however, requires frequent monitoring of the business applications and Extract to avoid business disruptions. Instead, Oracle recommends that you size the tablespace appropriately and set `AUTOEXTEND` to `ON` so that the tablespace does not fill up.

 **WARNING:**

Do not edit any other parameters in `params.sql` except if you need to follow documented instructions to change certain object names.

14. If you are changing the DDL schema for this installation, edit the `GLOBALS` file and specify the new schema name with the following parameter.

```
GGSCHEMA schema_name
```

15. Run the `marker_setup` script to reinstall the Oracle GoldenGate marker support system. You are prompted for the name of the Oracle GoldenGate schema.
16. Run the `ddl_setup` script. You are prompted for the name of the Oracle GoldenGate DDL schema.
17. Run the `role_setup` script to recreate the Oracle GoldenGate DDL role.
18. Grant the role to all Oracle GoldenGate users under which the following Oracle GoldenGate processes run: Extract, Replicat, GGSCI, and Manager. You might need to make multiple grants if the processes have different user names.
19. Run the `ddl_enable.sql` script to enable the DDL trigger.

Removing the DDL Objects from the System

This procedure removes the DDL environment and removes the history that maintains continuity between source and target DDL operations.



Note:

Due to object interdependencies, all objects must be removed.

1. Run GGSCI.
2. Stop Extract to stop DDL capture.

```
STOP EXTRACT group
```
3. Stop Replicat to stop DDL replication.

```
STOP REPLICAT group
```
4. Change directories to the Oracle GoldenGate installation directory.
5. Run SQL*Plus and log in as a user that has SYSDBA privileges.
6. Disconnect all sessions that ever issued DDL, including those of Oracle GoldenGate processes, SQL*Plus, business applications, and any other software that uses Oracle. Otherwise the database might generate an ORA-04021 error.
7. Run the `ddl_disable` script to disable the DDL trigger.
8. Run the `ddl_remove` script to remove the Oracle GoldenGate DDL trigger, the DDL history and marker tables, and the associated objects. This script produces a `ddl_remove_spool.txt` file that logs the script output and a `ddl_remove_set.txt` file that logs current user environment settings in case they are needed for debugging.
9. Run the `marker_remove` script to remove the Oracle GoldenGate marker support system. This script produces a `marker_remove_spool.txt` file that logs the script output and a `marker_remove_set.txt` file that logs environment settings in case they are needed for debugging.

Automatic Conflict Detection and Resolution

You can configure Oracle GoldenGate to automatically detect and resolve conflicts that occur when same data is updated concurrently at different sites.

Topics:

- [About Automatic Conflict Detection and Resolution](#)
- [Configuring Automatic Conflict Detection and Resolution](#)
You can configure Oracle GoldenGate automatic conflict detection and resolution in Oracle Database with the `DBMS_GOLDENGATE_ADM` package.
- [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.
- [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.

About 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 these databases. To do this, you must ensure that PL/SQL call is done at the source and the target databases. This feature is intended for use with bi-directional replication.

Note:

This chapter is for the automatic conflict detection and resolution feature that is specific to Oracle GoldenGate 12c (12.3.0.1) and Oracle Database 12c Release 2 (12.2) and later, which is configured in an Oracle Database. There is also a general Oracle GoldenGate feature for conflict detection and resolution, which is called Oracle GoldenGate conflict detection and resolution (CDR). Oracle GoldenGate CDR is configured in the Replicat parameter file.

You can configure only one of the following types of automatic conflict detection and resolution for a single table:

- The automatic conflict detection and resolution feature that is specific to Oracle Database 12c Release 2 (12.2)
- Oracle GoldenGate CDR
- [Automatic Conflict Detection and Resolution](#)
- [Requirements for Automatic Conflict Detection and Resolution](#)
Learn the requirements for automatic conflict detection and resolution (ACDR).

- [Latest Timestamp Conflict Detection and Resolution](#)
- [Delta Conflict Detection and Resolution](#)
- [Column Groups](#)

Automatic Conflict Detection and Resolution

You can configure automatic conflict detection and resolution in an Oracle GoldenGate configuration that replicates tables between Oracle Databases. To configure conflict detection and resolution for a table, call the `ADD_AUTO_CDR` procedure in the `DBMS_GOLDENGATE_ADM` package.

When Oracle GoldenGate captures changes that originated at an Oracle Database, each change is encapsulated in a row logical change record (LCR). A row LCR is a structured representation of a DML row change. Each row LCR includes the operation type, old column values, and new column values. Multiple row LCRs can be part of a single database transaction.

When more than one replica of a table allows changes to the table, a conflict can occur when a change is made to the same row in two different databases at nearly the same time. Oracle GoldenGate replicates changes using the row LCRs. It detects a conflict by comparing the old values in the row LCR for the initial change from the origin database with the current values of the corresponding table row at the destination database identified by the key columns. If any column value does not match, then there is a conflict.

After a conflict is detected, Oracle GoldenGate can resolve the conflict by overwriting values in the row with some values from the row LCR, ignoring the values in the row LCR, or computing a delta to update the row values.

Automatic conflict detection and resolution does not require application changes for the following reasons:

- Oracle Database automatically creates and maintains invisible timestamp columns.
- Inserts, updates, and deletes use the delete tombstone log table to determine if a row was deleted.
- LOB column conflicts can be detected.
- Oracle Database automatically configures supplemental logging on required columns.



Note:

If you use the classic Replicat on tables that have Automatic Change Detection and Resolution enabled, the Extract might abend with the OGG-10461 Failed to retrieve timestamp error. This is because the internal trigger that inserts the records into tombstone tables, only fires on user DMLs. A classic Replicat suppresses all the triggers from firing, which results in missing inserts on tombstone tables.

 **See Also:**

- *Oracle Database Utilities* for information about supplemental logging

Requirements for Automatic Conflict Detection and Resolution

Learn the requirements for automatic conflict detection and resolution (ACDR).

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.

In Oracle 12.2, the tables must have a primary key with a not null constraint on it. The table cannot have any unique keys. In Oracle 18.1 and up, the table must have a primary key with a not null constraint or a unique key with a not null constraint. The supplemental logging is also a requirement.

Integrated 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

Replicat needs to add the parameter `MAPINVISIBLECOLUMNS`

Latest Timestamp Conflict Detection and Resolution

When you run the `ADD_AUTO_CDR` procedure in the `DBMS_GOLDENGATE_ADM` package to configure a table for automatic Oracle GoldenGate conflict detection and resolution, a hidden timestamp column is added to the table. This hidden timestamp column records the time of a row change, and this information is used to detect and resolve conflicts.

When a row LCR is applied, a conflict can occur for an `INSERT`, `UPDATE`, or `DELETE` operation. The following table describes each type of conflict and how it is resolved.

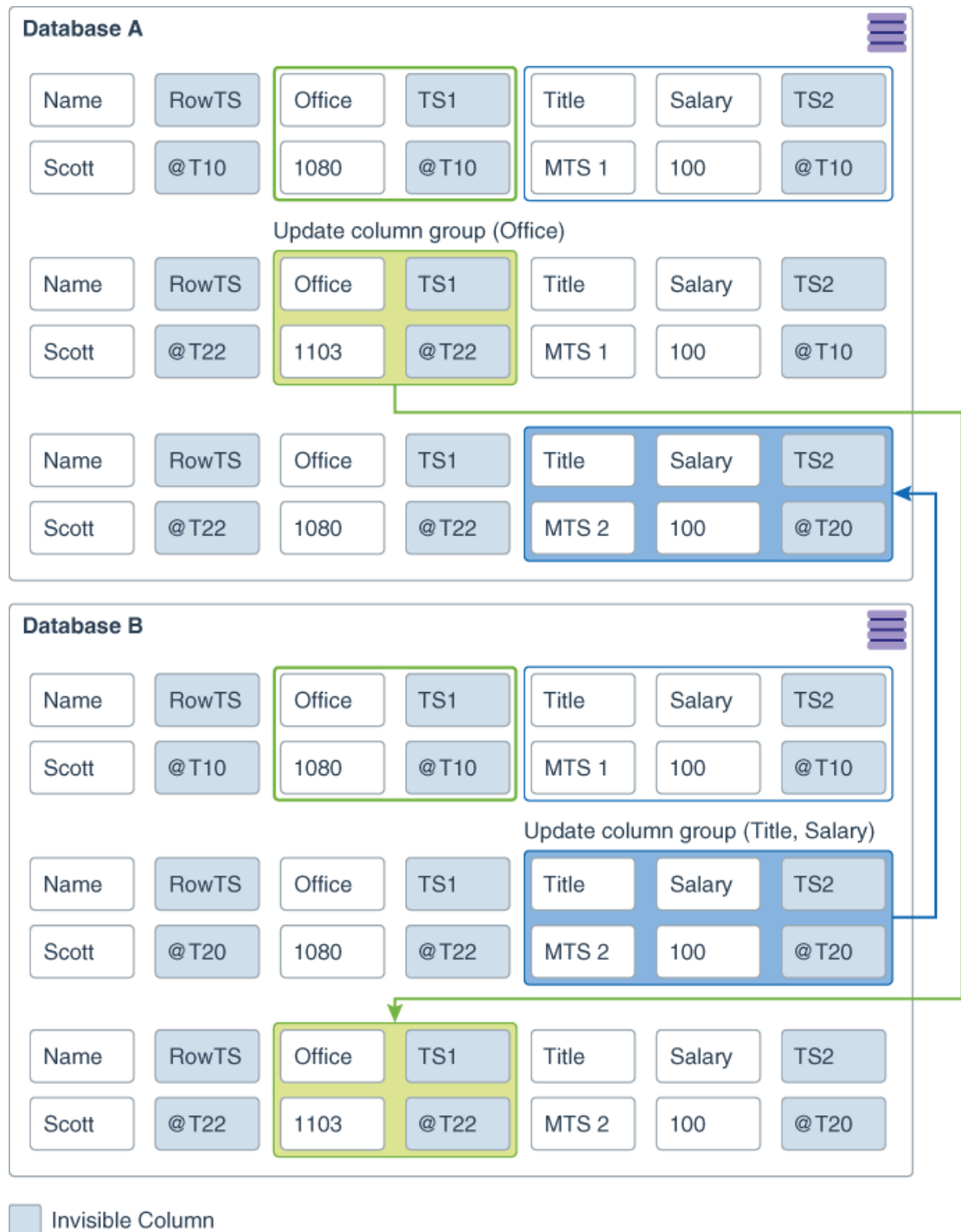
Operation	Conflict Detection	Conflict Resolution
INSERT	A conflict is detected when the table has the same value for a key column as the new value in the row LCR.	<p>If the timestamp of the row LCR is later than the timestamp in the table row, then the values in the row LCR replace the values in the table.</p> <p>If the timestamp of the row LCR is earlier than the timestamp in the table row, then the row LCR is discarded, and the table values are retained.</p>

Operation	Conflict Detection	Conflict Resolution
UPDATE	<p>A conflict is detected in each of the following cases:</p> <ul style="list-style-type: none"> • There is a mismatch between the timestamp value in the row LCR and the timestamp value of the corresponding row in the table. • There is a mismatch between an old value in a column group in the row LCR does not match the column value in the corresponding table row. A column group is a logical grouping of one or more columns in a replicated table. • The table row does not exist. If the row is in the tombstone table, then this is referred to as an update-delete conflict. 	<p>If there is a value mismatch and the timestamp of the row LCR is later than the timestamp in the table row, then the values in the row LCR replace the values in the table.</p> <p>If there is a value mismatch and the timestamp of the row LCR is earlier than the timestamp in the table row, then the row LCR is discarded, and the table values are retained.</p> <p>If the table row does not exist and the timestamp of the row LCR is later than the timestamp in the tombstone table row, then the row LCR is converted from an UPDATE operation to an INSERT operation and inserted into the table.</p> <p>If the table row does not exist and the timestamp of the row LCR is earlier than the timestamp in the tombstone table row, then the row LCR is discarded.</p> <p>If the table row does not exist and there is no corresponding row in the tombstone table, then the row LCR is converted from an UPDATE operation to an INSERT operation and inserted into the table.</p>
DELETE	<p>A conflict is detected in each of the following cases:</p> <ul style="list-style-type: none"> • There is a mismatch between the timestamp value in the row LCR and the timestamp value of the corresponding row in the table. • The table row does not exist. 	<p>If the timestamp of the row LCR is later than the timestamp in the table, then delete the row from the table.</p> <p>If the timestamp of the row LCR is earlier than the timestamp in the table, then the row LCR is discarded, and the table values are retained.</p> <p>If the delete is successful, then log the row LCR by inserting it into the tombstone table.</p> <p>If the table row does not exist, then log the row LCR by inserting it into the tombstone table.</p>

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.

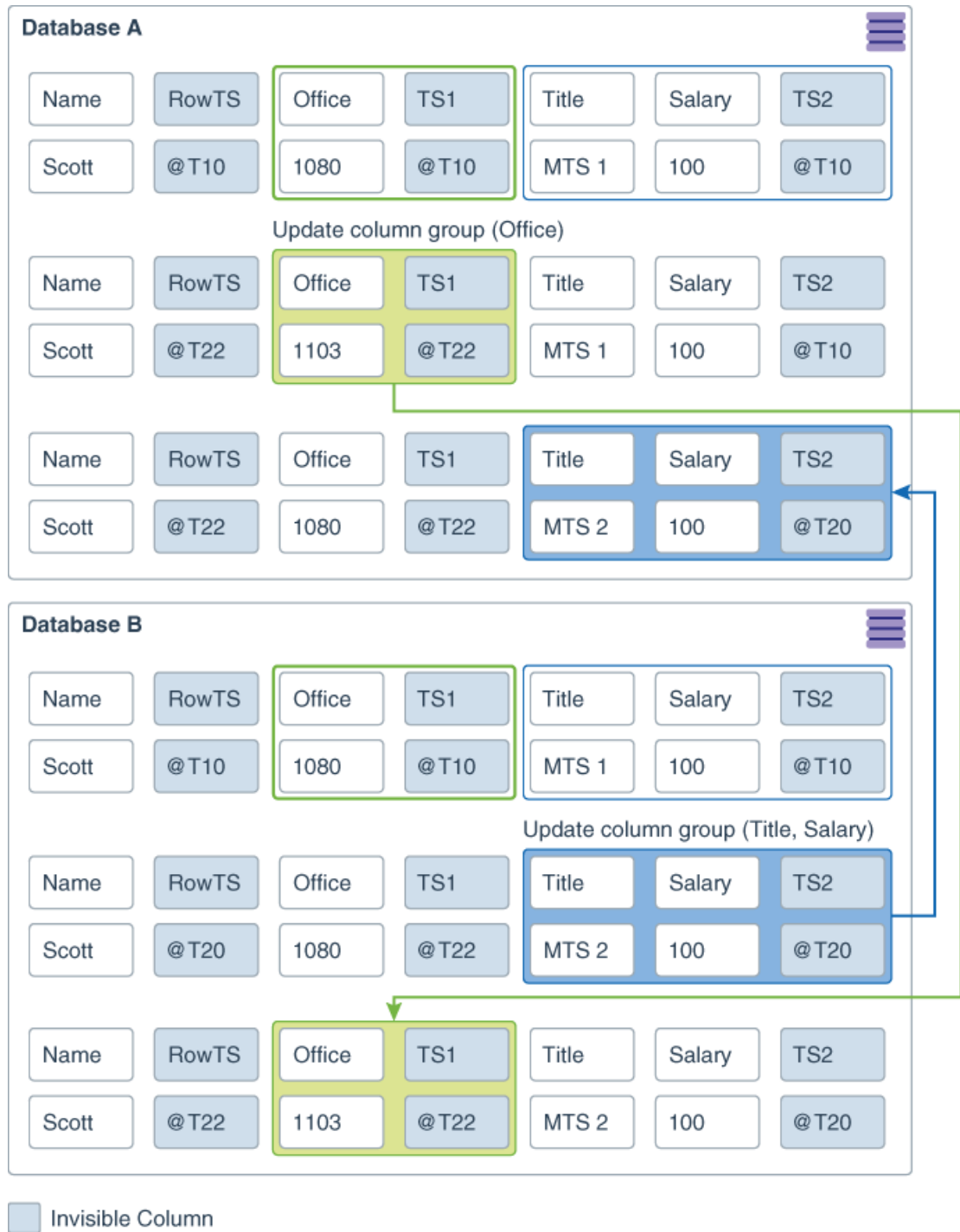
After delta conflict resolution, the value of the `Balance` column is the same for the row at database A and database B.

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

Configuring Automatic Conflict Detection and Resolution

You can configure Oracle GoldenGate automatic conflict detection and resolution in Oracle Database with the `DBMS_GOLDENGATE_ADM` package.

For the Replicat parameter file you need to add a `MAP` statement that includes the table to be replicated and the `MAPINVISIBLECOLUMNS` parameter.

- [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.
- [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.

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 14-1 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 14-2 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 14-3 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](#)
- [Altering a Column Group](#)
- [Purging Tombstone Rows](#)
- [Removing Conflict Detection and Resolution From a Table](#)
- [Removing a Column Group](#)

- [Removing Delta Conflict Detection and Resolution](#)

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 14-4 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 14-5 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 14-6 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 14-7 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 14-8 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 14-9 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](#)
- [Displaying Information About Conflict Resolution Columns](#)
- [Displaying Information About Column Groups](#)

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 14-10 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 14-11 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 14-12 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
  
```

15

Using Procedural Replication

Learn about procedural replication and how to configure it.

Topics:

- [About Procedural Replication](#)
- [Procedural Replication Process Overview](#)
Procedural replication uses a trail record to ensure that sufficient information is encapsulated with the record.
- [Enabling Procedural Replication](#)
Procedural replication is disabled by default. You can enable it by setting the `TRANLOGOPTIONS` option, `ENABLE_PROCEDURAL_REPLICATION`, to `yes`.
- [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.
- [Enabling and Disabling Supplemental Logging](#)
Oracle GoldenGate provides GGSCI commands to allow you to enable or disable procedural supplemental logging.
- [Filtering Features for Procedural Replication](#)
You can specify which procedures and packages you want to include or exclude for procedure replication.
- [Handling Procedural Replication Errors](#)
Procedural replication uses `REPERROR` parameter to configure the behavior of Replicat when an procedural error occurs.
- [Procedural Replication Pragma Options](#)
There are four pragma options for procedures: `AUTO`, `MANUAL`, `UNSUPPORTED`, and `NONE`.
- [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.
- [Monitoring Oracle GoldenGate Procedural Replication](#)
A set of data dictionary views enable you to monitor Oracle GoldenGate procedural replication.

About Procedural Replication

Oracle GoldenGate procedural replication is used 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.

Enabling Procedural Replication

Procedural replication is disabled by default. You can enable it by setting the `TRANLOGOPTIONS` option, `ENABLE_PROCEDURAL_REPLICATION`, to `yes`.

Once you enable the procedural option for one Extract, it remains on and can not be disabled.

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. Ensure that you are in triggerless mode, see [Prerequisites for Configuring DDL](#).
2. Connect to the source database as an Oracle GoldenGate administrator with `dblogin`.
3. Set the `TRANLOGOPTIONS` parameter option to `yes`.

```
TRANLOGOPTIONS INTEGRATEDPARAMS (ENABLE_PROCEDURAL_REPLICATION Y)
```

Procedural replication is enabled for Extract.

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 15-1 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 GGSCI 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 admin PASSWORD  
adminpw
```

```
DBLOGIN USERIDALIAS admin_dba DOMAIN OracleGoldenGate
```

2. Add supplemental logging for procedural replication.

```
ADD PROCEDURETRANDATA
```

```
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 admin PASSWORD  
adminpw
```

```
DBLOGIN USERIDALIAS admin_dba 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 admin PASSWORD  
adminpw
```

```
DBLOGIN USERIDALIAS admin_dba 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. See [Advanced Queue Concepts](#).

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 *Reference for Oracle GoldenGate* for other options.

```
REPERROR (PROCEDURE, DISCARD)
```

You have successfully handled errors for procedural replication.

Procedural Replication Pragma Options

There are four pragma options for procedures: `AUTO`, `MANUAL`, `UNSUPPORTED`, and `NONE`.

PL/SQL enter and exit markers are logged for procedures with pragmas `AUTO`, `MANUAL`, and `UNSUPPORTED`. The redo logs generated between the enter and exit markers are grouped and discarded.

Following is a list of the packages and procedures that are pragma constructs for replication. Any package or procedure not in this list is not considered a pragma construct for PL/SQL replication and is equivalent to pragma `NONE`.

PL/SQL Procedures with Pragma are UNSUPPORTED

Procedures and packages with the pragma `UNSUPPORTED` stop apply at the point of procedure invocation so that manual intervention can be taken. The following procedures are pragma and `UNSUPPORTED`.

Sche ma	Package	Procedure	Pragma
SYS	DBMS_REDEFINITION	ABORT_UPDATE	PRAGMA UNSUPPORTED
SYS	DBMS_REDEFINITION	EXECUTE_UPDATE	PRAGMA UNSUPPORTED
XDB	DBMS_XDBZ	ADD_APPLICATION_PRINCIPAL	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDBZ	CHANGE_APPLICATION_MEMBERSHIP	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDBZ	DELETE_APPLICATION_PRINCIPAL	PRAGMA UNSUPPORTED with COMMIT

Sche ma	Package	Procedure	Pragma
XDB	DBMS_XDBZ	SET_APPLICATION_P RINCIPAL	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_ADMIN	CREATENONCEKEY	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_ADMIN	INSTALLDEFAULTWAL LET	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_ADMIN	MOVEXDB_TABLESPAC E	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_ADMIN	REBUILDHIERARCHIC ALINDEX	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_CONFIG	ADDAUTHENTICATION MAPPING	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_CONFIG	ADDAUTHENTICATION METHOD	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_CONFIG	ADDTRUSTMAPPING	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_CONFIG	ADDTRUSTSCHEME	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_CONFIG	CLEARHTTDPDIGESTS	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_CONFIG	DELETEAUTHENTICAT IONMAPPING	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_CONFIG	DELETEAUTHENTICAT IONMETHOD	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_CONFIG	DELETETRUSTMAPPIN G	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_CONFIG	DELETETRUSTSCHEME	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_CONFIG	ENABLECUSTOMAUTHE NTICATION	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_CONFIG	ENABLECUSTOMTRUST	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_CONFIG	ENABLEDIGESTAUTHE NTICATION	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_CONFIG	ISGLOBALPORTENABL ED	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_CONFIG	SETDYNAMICGROUPST ORE	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_CONFIG	SETGLOBALPORTENAB LED	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XDB_CONFIG	SETHTTPCONFIGREAL M	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XMLINDEX	DROPPARAMETER	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XMLINDEX	MODIFYPARAMETER	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XMLINDEX	REGISTERPARAMETER	PRAGMA UNSUPPORTED with COMMIT
XDB	DBMS_XMLSCHEMA	COPYEVOLVE	PRAGMA UNSUPPORTED with COMMIT

PL/SQL Procedures with Pragma AUTO

For the procedures and packages with the pragma AUTO, the top-level PL/SQL API is called during apply.

Sche ma	Package	Procedure	Pragma
DVSY	DBMS_MACADM	ADD_AUTH_TO_REALM	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	ADD_CMD_RULE_TO_P OLICY	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	ADD_FACTOR_LINK	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	ADD_INDEX_FUNCATIO N	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	ADD-NLS_DATA	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	ADD_OBJECT_TO_REA LM	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	ADD_OWNER_TO_POLI CY	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	ADD_POLICY_FACTOR	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	ADD_REALM_TO_POLI CY	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	ADD_RULE_TO_RULE_ SET	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	AUTHORIZE_DATAPUM P_USER	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	AUTHORIZE_DDL	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	AUTHORIZE_DIAGNOS TIC_ADMIN	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	AUTHORIZE_MAINTEN ANCE_USER	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	AUTHORIZE_PREPROC ESSOR	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	AUTHORIZE_PROXY_U SER	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	AUTHORIZE_SCHEDUL ER_USER	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	AUTHORIZE_TTS_USE R	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	CHANGE_IDENTITY_F ACTOR	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	CHANGE_IDENTITY_V ALUE	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	CREATE_COMMAND_RU LE	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	CREATE_CONNECT_CO MMAND_RULE	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	CREATE_DOMAIN_IDE NTITY	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	CREATE_FACTOR	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	CREATE_FACTOR_TYP E	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	CREATE_IDENTITY	PRAGMA AUTO with COMMIT

Sche ma	Package	Procedure	Pragma
DVSY	DBMS_MACADM	CREATE_IDENTITY_M AP	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	CREATE_MAC_POLICY	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	CREATE_POLICY	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	CREATE_POLICY_LAB EL	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	CREATE_REALM	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	CREATE_ROLE	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	CREATE_RULE	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	CREATE_RULE_SET	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	CREATE_SESSION_EV ENT_CMD_RULE	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	CREATE_SESSION_EV ENT_CMD_RULE	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_AUTH_FROM_ REALM	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_CMD_RULE_F ROM_POLICY	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_COMMAND_RU LE	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_CONNECT_CO MMAND_RULE	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_FACTOR	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_FACTOR_LIN K	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_FACTOR_TYP E	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_IDENTITY	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_IDENTITY_M AP	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_INDEX_FUNC TION	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_MAC_POLICY _CASCADE	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_OBJECT_FRO M_REALM	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_OWNER_FROM _POLICY	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_POLICY_FAC TOR	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_POLICY_LAB EL	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_REALM	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_REALM_CASC ADE	PRAGMA AUTO with COMMIT

Sche ma	Package	Procedure	Pragma
DVSY	DBMS_MACADM	DELETE_REALM_FROM_POLICY	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_ROLE	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_RULE	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_RULE_FROM_RULE_SET	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_RULE_SET	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_SESSION_EVENT_CMD_RULE	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DELETE_SYSTEM_EVENT_CMD_RULE	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DISABLE_DV	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DISABLE_DV_DICTIONARY_ACCTS	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DISABLE_DV_PATCH_ADMIN_AUDIT	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DISABLE_ORADEBUG	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DROP_DOMAIN_IDENTITY	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	DROP_POLICY	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	ENABLE_DV	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	ENABLE_DV_DICTIONARY_ACCTS	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	ENABLE_DV_PATCH_ADMIN_AUDIT	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	ENABLE_ORADEBUG	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	RENAME_FACTOR	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	RENAME_FACTOR_TYPE	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	RENAME_POLICY	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	RENAME_REALM	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	RENAME_ROLE	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	RENAME_RULE	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	RENAME_RULE_SET	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	UNAUTHORIZE_DATAPUMP_USER	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	UNAUTHORIZE_DDL	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	UNAUTHORIZE_DIAGNOSTIC_ADMIN	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	UNAUTHORIZE_MAINTENANCE_USER	PRAGMA AUTO with COMMIT
DVSY	DBMS_MACADM	UNAUTHORIZE_PREPROCESSOR	PRAGMA AUTO with COMMIT

Sche ma	Package	Procedure	Pragma
DVSYs	DBMS_MACADM	UNAUTHORIZE_PROXY_USER	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	UNAUTHORIZE_SCHEDULER_USER	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	UNAUTHORIZE_TTS_USER	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	UPDATE_COMMAND_RULE	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	UPDATE_CONNECT_COMMAND_RULE	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	UPDATE_FACTOR	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	UPDATE_FACTOR_TYPE	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	UPDATE_IDENTITY	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	UPDATE_MAC_POLICY	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	UPDATE_POLICY_DESCRIPTION	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	UPDATE_POLICY_STATE	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	UPDATE_REALM	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	UPDATE_REALM_AUTH	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	UPDATE_ROLE	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	UPDATE_RULE	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	UPDATE_RULE_SET	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	UPDATE_SESSION_EVENT_CMD_RULE	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	UPDATE_SYSTEM_EVENT_CMD_RULE	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	CREATE_ADMIN_AUDIT	PRAGMA AUTO
DVSYs	DBMS_MACADM	CREATE_MACOLS_CONTEXTS	PRAGMA AUTO with COMMIT
DVSYs	DBMS_MACADM	DROP_MACOLS_CONTEXTS	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_EVENTS	AFTER_CREATE	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_EVENTS	AFTER_DROP	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_EVENTS	BEFORE_ALTER	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_LGSTNDBY_UTIL	ADD_COMPARTMENTS	PRAGMA AUTO
LBACS YS	LBAC_LGSTNDBY_UTIL	ADD_GROUPS	PRAGMA AUTO

Sche ma	Package	Procedure	Pragma
LBACS YS	LBAC_LGSTNDBY_U TIL	ALTER_COMPARTMENT S	PRAGMA AUTO
LBACS YS	LBAC_LGSTNDBY_U TIL	ALTER_GROUPS	PRAGMA AUTO
LBACS YS	LBAC_LGSTNDBY_U TIL	CONFIGURE_OLS	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_LGSTNDBY_U TIL	CREATE_POLICY	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_LGSTNDBY_U TIL	DISABLE_OLS	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_LGSTNDBY_U TIL	DROP_ALL_COMPARTM ENTS	PRAGMA AUTO
LBACS YS	LBAC_LGSTNDBY_U TIL	DROP_ALL_GROUPS	PRAGMA AUTO
LBACS YS	LBAC_LGSTNDBY_U TIL	DROP_COMPARTMENTS	PRAGMA AUTO
LBACS YS	LBAC_LGSTNDBY_U TIL	DROP_GROUPS	PRAGMA AUTO
LBACS YS	LBAC_LGSTNDBY_U TIL	ENABLE_OLS	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_LGSTNDBY_U TIL	INSERT_LABEL	PRAGMA AUTO
LBACS YS	LBAC_LGSTNDBY_U TIL	SAVE_DEFAULT_LABE LS	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_LGSTNDBY_U TIL	SET_COMPARTMENTS	PRAGMA AUTO
LBACS YS	LBAC_LGSTNDBY_U TIL	SET_DEFAULT_LABEL	PRAGMA AUTO
LBACS YS	LBAC_LGSTNDBY_U TIL	SET_GROUPS	PRAGMA AUTO
LBACS YS	LBAC_LGSTNDBY_U TIL	SET_LEVELS	PRAGMA AUTO
LBACS YS	LBAC_LGSTNDBY_U TIL	SET_ROW_LABEL	PRAGMA AUTO
LBACS YS	LBAC_LGSTNDBY_U TIL	SET_USER_LABELS	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_LGSTNDBY_U TIL	STORE_LABEL_LIST	PRAGMA AUTO
LBACS YS	LBAC_POLICY_ADM IN	ALTER_SCHEMA_POLI CY	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_POLICY_ADM IN	APPLY_SCHEMA_POLI CY	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_POLICY_ADM IN	APPLY_TABLE_POLIC Y	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_POLICY_ADM IN	DISABLE_SCHEMA_PO LICY	PRAGMA AUTO with COMMIT

Sche ma	Package	Procedure	Pragma
LBACS YS	LBAC_POLICY_ADM IN	DISABLE_TABLE_POL ICY	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_POLICY_ADM IN	ENABLE_SCHEMA_POL ICY	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_POLICY_ADM IN	ENABLE_TABLE_POLI CY	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_POLICY_ADM IN	POLICY_SUBSCRIBE	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_POLICY_ADM IN	POLICY_UNSUBSCRIB E	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_POLICY_ADM IN	REMOVE_SCHEMA_POL ICY	PRAGMA AUTO with COMMIT
LBACS YS	LBAC_POLICY_ADM IN	REMOVE_TABLE_POLI CY	PRAGMA AUTO with COMMIT
LBACS YS	SA_AUDIT_ADMIN	AUDIT	PRAGMA AUTO with COMMIT
LBACS YS	SA_AUDIT_ADMIN	AUDIT_LABEL	PRAGMA AUTO with COMMIT
LBACS YS	SA_AUDIT_ADMIN	AUDIT_LABEL_ENABL ED	PRAGMA AUTO with COMMIT
LBACS YS	SA_AUDIT_ADMIN	AUDIT_LABEL_ENABL ED_SQL	PRAGMA AUTO with COMMIT
LBACS YS	SA_AUDIT_ADMIN	CREATE_VIEW	PRAGMA AUTO with COMMIT
LBACS YS	SA_AUDIT_ADMIN	DROP_VIEW	PRAGMA AUTO with COMMIT
LBACS YS	SA_AUDIT_ADMIN	NOAUDIT	PRAGMA AUTO with COMMIT
LBACS YS	SA_AUDIT_ADMIN	NOAUDIT_LABEL	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	ALTER_COMPARTMENT	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	ALTER_COMPARTMENT	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	ALTER_GROUP	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	ALTER_GROUP	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	ALTER_GROUP_PAREN T	PRAGMA AUTO
LBACS YS	SA_COMPONENTS	ALTER_GROUP_PAREN T	PRAGMA AUTO
LBACS YS	SA_COMPONENTS	ALTER_GROUP_PAREN T	PRAGMA AUTO
LBACS YS	SA_COMPONENTS	ALTER_LEVEL	PRAGMA AUTO with COMMIT

Sche ma	Package	Procedure	Pragma
LBACS YS	SA_COMPONENTS	ALTER_LEVEL	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	CREATE_COMPARTMEN T	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	CREATE_GROUP	PRAGMA AUTO
LBACS YS	SA_COMPONENTS	CREATE_LEVEL	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	DROP_COMPARTMENT	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	DROP_COMPARTMENT	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	DROP_GROUP	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	DROP_GROUP	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	DROP_LEVEL	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	DROP_LEVEL	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	ALTER_LABEL	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	ALTER_LABEL	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	CREATE_LABEL	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	DROP_LABEL	PRAGMA AUTO with COMMIT
LBACS YS	SA_COMPONENTS	DROP_LABEL	PRAGMA AUTO with COMMIT
LBACS YS	SA_SYSDBA	ALTER_POLICY	PRAGMA AUTO with COMMIT
LBACS YS	SA_SYSDBA	DISABLE_POLICY	PRAGMA AUTO with COMMIT
LBACS YS	SA_SYSDBA	DROP_POLICY	PRAGMA AUTO with COMMIT
LBACS YS	SA_SYSDBA	ENABLE_POLICY	PRAGMA AUTO with COMMIT
LBACS YS	SA_USER_ADMIN	DROP_USER_ACCESS	PRAGMA AUTO with COMMIT
LBACS YS	SA_USER_ADMIN	SET_PROG_PRIVS	PRAGMA AUTO with COMMIT
LBACS YS	SA_USER_ADMIN	SET_USER_PRIVS	PRAGMA AUTO with COMMIT
SYS	DBMS_AQ	AQ\$_BACKGROUND_OP ER	PRAGMA AUTO

Sche ma	Package	Procedure	Pragma
SYS	DBMS_AQ	AQ\$_DELETE_DIOT_T AB	PRAGMA AUTO
SYS	DBMS_AQ	AQ\$_DELETE_HIST_T AB	PRAGMA AUTO
SYS	DBMS_AQ	AQ\$_DELETE_TIOT_T AB	PRAGMA AUTO
SYS	DBMS_AQ	AQ\$_INSERT_DIOT_T AB	PRAGMA AUTO
SYS	DBMS_AQ	AQ\$_INSERT_HIST_T AB	PRAGMA AUTO
SYS	DBMS_AQ	AQ\$_INSERT_TIOT_T AB	PRAGMA AUTO
SYS	DBMS_AQ	AQ\$_UPDATE_HIST_T AB	PRAGMA AUTO
SYS	DBMS_AQ	AQ\$_UPDATE_HIST_T AB_EX	PRAGMA AUTO
SYS	DBMS_AQ	DEQUEUE_INTERNAL	PRAGMA AUTO
SYS	DBMS_AQ	ENQUEUE_INT_SHARD	PRAGMA AUTO
SYS	DBMS_AQ	ENQUEUE_INT_SHARD	PRAGMA AUTO
SYS	DBMS_AQ	ENQUEUE_INT_SHARD	PRAGMA AUTO
SYS	DBMS_AQ	ENQUEUE_INT_SHARD _JMS	PRAGMA AUTO
SYS	DBMS_AQ	ENQUEUE_INT_UNSHA RDED	PRAGMA AUTO
SYS	DBMS_AQ	ENQUEUE_INT_UNSHA RDED	PRAGMA AUTO
SYS	DBMS_AQ	ENQUEUE_INT_UNSHA RDED	PRAGMA AUTO
SYS	DBMS_AQ	ENQUEUE_INT_UNSHA RDED	PRAGMA AUTO
SYS	DBMS_AQ	REGISTRATION_REPL ICATION	PRAGMA AUTO
SYS	DBMS_AQADM	ALTER_AQ_AGENT	PRAGMA AUTO
SYS	DBMS_AQADM	CREATE_AQ_AGENT	PRAGMA AUTO
SYS	DBMS_AQADM	DISABLE_DB_ACCESS	PRAGMA AUTO
SYS	DBMS_AQADM	DROP_AQ_AGENT	PRAGMA AUTO
SYS	DBMS_AQADM	ENABLE_DB_ACCESS	PRAGMA AUTO
SYS	DBMS_AQADM	GRANT_SYSTEM_PRIV ILEGE	PRAGMA AUTO
SYS	DBMS_AQADM	GRANT_TYPE_ACCESS	PRAGMA AUTO
SYS	DBMS_AQADM	REVOKE_SYSTEM_PRI VILEGE	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	ALTER_QUEUE	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	ALTER_QUEUE_TABLE	PRAGMA AUTO

Sche ma	Package	Procedure	Pragma
SYS	DBMS_AQADM_SYS	ALTER_SHARDED_QUE UE	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	ALTER_SUBSCRIBER_ 11G	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	CREATE_EVICTIION_T ABLE	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	CREATE_EXCEPTION_ QUEUE	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	CREATE_NP_QUEUE_I NT	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	CREATE_QUEUE	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	CREATE_QUEUE_TABL E	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	CREATE_SHARDED_QU EUE	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	DROP_EVICTIION_TAB LE	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	DROP_QUEUE	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	DROP_QUEUE_TABLE	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	DROP_SHARDED_QUEU E_INT	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	ENABLE_JMS_TYPES_ INT	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	GRANT_QUEUE_PRIVI LEGE	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	MIGRATE_QUEUE_TAB LE	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	PATCH_QUEUE_TABLE	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	PATCH_QUEUE_TABLE	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	PSTUPD_CREATE_EVI CTION_TABLE	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	PURGE_QUEUE_TABLE _INT	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	REMOVE_ORPHMSG_I NT	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	REMOVE_SUBSCRIBER _11G_INT	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	REVOKE_QUEUE_PRIV ILEGE	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	START_QUEUE	PRAGMA AUTO
SYS	DBMS_AQADM_SYS	STOP_QUEUE	PRAGMA AUTO
SYS	DBMS_AQELM	SET_MAILHOST	PRAGMA AUTO
SYS	DBMS_AQELM	SET_MAILPORT	PRAGMA AUTO
SYS	DBMS_AQELM	SET_PROXY	PRAGMA AUTO

Sche ma	Package	Procedure	Pragma
SYS	DBMS_AQELM	SET_SENDFROM	PRAGMA AUTO
SYS	DBMS_AQ_SYS_IMP _INTERNAL	BUMP_TID_SEQUENCE	PRAGMA AUTO
SYS	DBMS_AQ_SYS_IMP _INTERNAL	CLEANUP_SCHEMA_IM PORT	PRAGMA AUTO
SYS	DBMS_AQ_SYS_IMP _INTERNAL	IMPORT_CMT_TIME_T ABLE	PRAGMA AUTO with COMMIT
SYS	DBMS_AQ_SYS_IMP _INTERNAL	IMPORT_DEQUEUELOG _TABLE	PRAGMA AUTO with COMMIT
SYS	DBMS_AQ_SYS_IMP _INTERNAL	IMPORT_EXP_ENTRY	PRAGMA AUTO
SYS	DBMS_AQ_SYS_IMP _INTERNAL	IMPORT_HISTORY_TA BLE	PRAGMA AUTO with COMMIT
SYS	DBMS_AQ_SYS_IMP _INTERNAL	IMPORT_INDEX_TABL E	PRAGMA AUTO with COMMIT
SYS	DBMS_AQ_SYS_IMP _INTERNAL	IMPORT_QTAB_EXPDE P	PRAGMA AUTO
SYS	DBMS_AQ_SYS_IMP _INTERNAL	IMPORT_QUEUE	PRAGMA AUTO with COMMIT
SYS	DBMS_AQ_SYS_IMP _INTERNAL	IMPORT_QUEUE_META	PRAGMA AUTO
SYS	DBMS_AQ_SYS_IMP _INTERNAL	IMPORT_QUEUE_SEQ	PRAGMA AUTO
SYS	DBMS_AQ_SYS_IMP _INTERNAL	IMPORT_QUEUE_TABL E	PRAGMA AUTO with COMMIT
SYS	DBMS_AQ_SYS_IMP _INTERNAL	IMPORT_SIGNATURE_ TABLE	PRAGMA AUTO with COMMIT
SYS	DBMS_AQ_SYS_IMP _INTERNAL	IMPORT_SUBSCRIBER _TABLE	PRAGMA AUTO with COMMIT
SYS	DBMS_AQ_SYS_IMP _INTERNAL	IMPORT_TIMEMGR_TA BLE	PRAGMA AUTO with COMMIT
SYS	DBMS_AQ_SYS_IMP _INTERNAL	POST_TTS_REBUILD_ IDX	PRAGMA AUTO with COMMIT
SYS	DBMS_AQ_SYS_IMP _INTERNAL	POST_TTS_SHARDED_ Q	PRAGMA AUTO
SYS	DBMS_AQ_SYS_IMP _INTERNAL	POST_TTS_WORK	PRAGMA AUTO
SYS	DBMS_AQ_SYS_IMP _INTERNAL	POST_TTS_WORK_REM AINING	PRAGMA AUTO
SYS	DBMS_DBFS_CONTE NT_ADMIN	EXIM_MOUNT	PRAGMA AUTO
SYS	DBMS_DBFS_CONTE NT_ADMIN	EXIM_MOUNTP	PRAGMA AUTO
SYS	DBMS_DBFS_CONTE NT_ADMIN	EXIM_STORE	PRAGMA AUTO

Sche ma	Package	Procedure	Pragma
SYS	DBMS_DBFS_CONTE NT_ADMIN	MOUNTSTORE_LOG	PRAGMA AUTO
SYS	DBMS_DBFS_CONTE NT_ADMIN	REGISTERSTORE_LOG	PRAGMA AUTO
SYS	DBMS_DBFS_CONTE NT_ADMIN	UNMOUNTSTORE_LOG	PRAGMA AUTO
SYS	DBMS_DBFS_CONTE NT_ADMIN	UNREGISTERSTORE_L OG	PRAGMA AUTO
SYS	DBMS_DBFS_SFS	NORMALIZEFS	PRAGMA AUTO with COMMIT
SYS	DBMS_DBFS_CONTE NT_ADMIN	REORGANIZEFS	PRAGMA AUTO with COMMIT
SYS	DBMS_DBFS_CONTE NT_ADMIN	SHRINKFS	PRAGMA AUTO with COMMIT
SYS	DBMS_DBFS_SFS_A DMIN	CREATEFILESYSTEM_ LOG	PRAGMA AUTO
SYS	DBMS_DBFS_SFS_A DMIN	DELETE_ORPHANS_LO G	PRAGMA AUTO with COMMIT
SYS	DBMS_DBFS_SFS_A DMIN	DROFFILESYSTEM_LO G	PRAGMA AUTO
SYS	DBMS_DBFS_SFS_A DMIN	EXIM_ATTRV	PRAGMA AUTO with COMMIT
SYS	DBMS_DBFS_SFS_A DMIN	EXIM_FS	PRAGMA AUTO
SYS	DBMS_DBFS_SFS_A DMIN	EXIM_GRANTS	PRAGMA AUTO with COMMIT
SYS	DBMS_DBFS_SFS_A DMIN	EXIM_SEQ	PRAGMA AUTO
SYS	DBMS_DBFS_SFS_A DMIN	EXIM_SNAP	PRAGMA AUTO
SYS	DBMS_DBFS_SFS_A DMIN	EXIM_TABP	PRAGMA AUTO
SYS	DBMS_DBFS_SFS_A DMIN	EXIM_TAB_LOG	PRAGMA AUTO
SYS	DBMS_DBFS_SFS_A DMIN	EXIM_VOL	PRAGMA AUTO
SYS	DBMS_DBFS_SFS_A DMIN	INITFILESYSTEM_LO G	PRAGMA AUTO
SYS	DBMS_DBFS_SFS_A DMIN	PARTITION_SEQUENC E_LOG	PRAGMA AUTO with COMMIT
SYS	DBMS_DBFS_SFS_A DMIN	RECACHE_SEQUENCE_ LOG	PRAGMA AUTO with COMMIT
SYS	DBMS_DBFS_SFS_A DMIN	REGISTERFILESYSTE M_LOG	PRAGMA AUTO
SYS	DBMS_DBFS_SFS_A DMIN	SETFSPROPERTIES_L OG	PRAGMA AUTO

Sche ma	Package	Procedure	Pragma
SYS	DBMS_DBFS_SFS_A DMIN	UNREGISTERFILESYS TEM_LOG	PRAGMA AUTO
SYS	DBMS_DDL	SET_TRIGGER_FIRIN G_PROPERTY	PRAGMA AUTO with COMMIT
SYS	DBMS_DDL	SET_TRIGGER_FIRIN G_PROPERTY	PRAGMA AUTO with COMMIT
SYS	DBMS_FGA	ADD_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_FGA	DISABLE_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_FGA	DROP_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_FGA	ENABLE_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_GOLDENGATE _ADM_INT_I	ADD_AUTO_CDR_COLG ROUP_INT	PRAGMA AUTO with COMMIT
SYS	DBMS_FGA	ADD_AUTO_CDR_DELT A_RES_INT	PRAGMA AUTO with COMMIT
SYS	DBMS_FGA	ADD_AUTO_CDR_INT	PRAGMA AUTO with COMMIT
SYS	DBMS_FGA	ALTER_AUTO_CDR_CO LGROUP_INT	PRAGMA AUTO with COMMIT
SYS	DBMS_FGA	ALTER_AUTO_CDR_IN T	PRAGMA AUTO with COMMIT
SYS	DBMS_FGA	REMOVE_AUTO_CDR_C OLGROUP_INT	PRAGMA AUTO with COMMIT
SYS	DBMS_FGA	REMOVE_AUTO_CDR_D ELTA_RES_INT	PRAGMA AUTO with COMMIT
SYS	DBMS_FGA	REMOVE_AUTO_CDR_I NT	PRAGMA AUTO with COMMIT
SYS	DBMS_GOLDENGATE _IMP	ACDR_COLUMN	PRAGMA AUTO with COMMIT
SYS	DBMS_GOLDENGATE _IMP	ACDR_COLUMN_GROUP	PRAGMA AUTO with COMMIT
SYS	DBMS_GOLDENGATE _IMP	ACDR_END	PRAGMA AUTO with COMMIT
SYS	DBMS_GOLDENGATE _IMP	ACDR_START	PRAGMA AUTO with COMMIT
SYS	DBMS_GOLDENGATE _IMP	ACDR_TABLE	PRAGMA AUTO with COMMIT
SYS	DBMS_INTERNAL_L OGSTDBY	EDS_EVOLVE_DISABL E	PRAGMA AUTO with COMMIT
SYS	DBMS_INTERNAL_L OGSTDBY	EDS_EVOLVE_ENABLE	PRAGMA AUTO with COMMIT
SYS	DBMS_INTERNAL_R OLLING	DESTROY_META	PRAGMA AUTO
SYS	DBMS_INTERNAL_R OLLING	INSERT_DGLRDDIR	PRAGMA AUTO
SYS	DBMS_INTERNAL_R OLLING	INSERT_DGLRDEVT	PRAGMA AUTO

Sche ma	Package	Procedure	Pragma
SYS	DBMS_INTERNAL_R OLLING	SET_UPGRADE_FLAGS	PRAGMA AUTO
SYS	DBMS_INTERNAL_R OLLING	UPDATE_DGLRDINS_P ROGRESS	PRAGMA AUTO
SYS	DBMS_INTERNAL_R OLLING	UPSERT_DGLRDCON	PRAGMA AUTO
SYS	DBMS_INTERNAL_R OLLING	UPSERT_DGLRDDAT	PRAGMA AUTO
SYS	DBMS_INTERNAL_R OLLING	UPSERT_DGLRDINS	PRAGMA AUTO
SYS	DBMS_INTERNAL_R OLLING	UPSERT_DGLRDPAR	PRAGMA AUTO
SYS	DBMS_INTERNAL_R OLLING	UPSERT_DGLRDSTA	PRAGMA AUTO
SYS	DBMS_INTERNAL_R OLLING	UPSERT_DGLRDSTS	PRAGMA AUTO
SYS	DBMS_ISCHED	CREATE_CREDENTIAL	PRAGMA AUTO with COMMIT
SYS	DBMS_ISCHED	EXEC_JOB_RUN_LSA	PRAGMA AUTO
SYS	DBMS_ISCHED	SET_AGENT_REGISTR ATION_PASS	PRAGMA AUTO with COMMIT
SYS	DBMS_PRVTAQIS	SUBID_REPLICATE	PRAGMA AUTO with COMMIT
SYS	DBMS_PRVTAQIS	ADD_DURABLE_SUB	PRAGMA AUTO with COMMIT
SYS	DBMS_PRVTAQIS	ALTER_SUBSCRIBER_ 12G	PRAGMA AUTO
SYS	DBMS_PRVTAQIS	REMOVE_SUBSCRIBER _12G	PRAGMA AUTO
SYS	DBMS_REDACT	ADD_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_REDACT	ALTER_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_REDACT	APPLY_POLICY_EXPR _TO_COL	PRAGMA AUTO with COMMIT
SYS	DBMS_REDACT	CREATE_POLICY_EXP RESSION	PRAGMA AUTO with COMMIT
SYS	DBMS_REDACT	DISABLE_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_REDACT	DROP_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_REDACT	DROP_POLICY_EXPRE SSION	PRAGMA AUTO with COMMIT
SYS	DBMS_REDACT	ENABLE_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_REDACT	FPM_MASK	PRAGMA AUTO with COMMIT
SYS	DBMS_REDACT	FPM_UNMASK	PRAGMA AUTO with COMMIT
SYS	DBMS_REDACT	UPDATE_FULL_REDAC TION_VALUES	PRAGMA AUTO with COMMIT
SYS	DBMS_REDACT	UPDATE_POLICY_EXP RESSION	PRAGMA AUTO with COMMIT

Sche ma	Package	Procedure	Pragma
SYS	DBMS_REDEFINITION	ABORT_REDEF_TABLE	PRAGMA AUTO with COMMIT
SYS	DBMS_REDEFINITION	ABORT_ROLLBACK	PRAGMA AUTO with COMMIT
SYS	DBMS_REDEFINITION	COPY_TABLE_DEPENDENTS	PRAGMA AUTO with COMMIT
SYS	DBMS_REDEFINITION	FINISH_REDEF_TABLE	PRAGMA AUTO with COMMIT
SYS	DBMS_REDEFINITION	REGISTER_DEPENDENT_OBJECT	PRAGMA AUTO with COMMIT
SYS	DBMS_REDEFINITION	ROLLBACK	PRAGMA AUTO with COMMIT
SYS	DBMS_REDEFINITION	SET_PARAM	PRAGMA AUTO with COMMIT
SYS	DBMS_REDEFINITION	START_REDEF_TABLE	PRAGMA AUTO with COMMIT
SYS	DBMS_REDEFINITION	SYNC_INTERIM_TABLE	PRAGMA AUTO with COMMIT
SYS	DBMS_REDEFINITION	UNREGISTER_DEPENDENT_OBJECT	PRAGMA AUTO with COMMIT
SYS	DBMS_RLS_INT	ADD_GROUPED_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_RLS_INT	ADD_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_RLS_INT	ADD_POLICY_CONTEXT	PRAGMA AUTO with COMMIT
SYS	DBMS_RLS_INT	ALTER_GROUPED_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_RLS_INT	ALTER_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_RLS_INT	CREATE_POLICY_GROUP	PRAGMA AUTO with COMMIT
SYS	DBMS_RLS_INT	DELETE_POLICY_GROUP	PRAGMA AUTO with COMMIT
SYS	DBMS_RLS_INT	DISABLE_GROUPED_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_RLS_INT	DROP_GROUPED_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_RLS_INT	DROP_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_RLS_INT	DROP_POLICY_CONTEXT	PRAGMA AUTO with COMMIT
SYS	DBMS_RLS_INT	ENABLE_GROUPED_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_RLS_INT	ENABLE_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_RLS_INT	REFRESH_GROUPED_POLICY	PRAGMA AUTO with COMMIT
SYS	DBMS_RLS_INT	REFRESH_POLICY	PRAGMA AUTO with COMMIT

Sche ma	Package	Procedure	Pragma
SYS	DBMS_RULEADM_IN TERNAL	ADD_RULE	PRAGMA AUTO
SYS	DBMS_RULEADM_IN TERNAL	ALTER_EVALUATION_ CONTEXT	PRAGMA AUTO
SYS	DBMS_RULEADM_IN TERNAL	ALTER_RULE	PRAGMA AUTO
SYS	DBMS_RULEADM_IN TERNAL	CREATE_EVALUATION_ CONTEXT	PRAGMA AUTO
SYS	DBMS_RULEADM_IN TERNAL	CREATE_RULE	PRAGMA AUTO
SYS	DBMS_RULEADM_IN TERNAL	CREATE_RULE_SET	PRAGMA AUTO
SYS	DBMS_RULEADM_IN TERNAL	DROP_EVALUATION_C ONTEXT	PRAGMA AUTO
SYS	DBMS_RULEADM_IN TERNAL	DROP_RULE	PRAGMA AUTO
SYS	DBMS_RULEADM_IN TERNAL	DROP_RULE_SET	PRAGMA AUTO
SYS	DBMS_RULEADM_IN TERNAL	REMOVE_RULE	PRAGMA AUTO
SYS	DBMS_RULE_ADM	GRANT_OBJECT_PRIV ILEGE	PRAGMA AUTO
SYS	DBMS_RULE_ADM	GRANT_SYSTEM_PRIV ILEGE	PRAGMA AUTO
SYS	DBMS_RULE_ADM	REVOKE_OBJECT_PRI VILEGE	PRAGMA AUTO
SYS	DBMS_RULE_ADM	REVOKE_SYSTEM_PRI VILEGE	PRAGMA AUTO
SYS	DBMS_SCHEDULER	ADD_EVENT_QUEUE_S UBSCRIBER	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	ADD_GROUP_MEMBER	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	ADD_JOB_EMAIL_NOT IFICATION	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	ADD_TO_INCOMPATIB ILITY	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	ADD_WINDOW_GROUP_ MEMBER	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	ALTER_CHAIN	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	ALTER_CHAIN	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	ALTER_RUNNING_CHA IN	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	ALTER_RUNNING_CHA IN	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	ANALYZE_CHAIN	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	AUTO_PURGE	PRAGMA AUTO with COMMIT

Sche ma	Package	Procedure	Pragma
SYS	DBMS_SCHEDULER	CHECK_CREDENTIAL	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	COPY_JOB	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_CHAIN	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_DATABASE_D ESTINATION	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_EVENT_SCHE DULE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_FILE_WATCH ER	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_GROUP	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_INCOMPATIB ILITY	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_JOB	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_JOB	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_JOB	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_JOB	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_JOB	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_JOB	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_JOBS	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_JOBS	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_JOB_CLASS	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_PROGRAM	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_RESOURCE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_SCHEDULE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_WINDOW	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_WINDOW	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	CREATE_WINDOW_GRO UP	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DEFINE_ANYDATA_AR GUMENT	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DEFINE_CHAIN_EVEN T_STEP	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DEFINE_CHAIN_EVEN T_STEP	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DEFINE_CHAIN_RULE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DEFINE_CHAIN_STEP	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DEFINE_METADATA_A RGUMENT	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DEFINE_PROGRAM_AR GUMENT	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DEFINE_PROGRAM_AR GUMENT	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DELETE_FILE	PRAGMA AUTO with COMMIT

Sche ma	Package	Procedure	Pragma
SYS	DBMS_SCHEDULER	DISABLE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DISABLE1_CALENDAR _CHECK	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_AGENT_DESTIN ATION	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_CHAIN	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_CHAIN_RULE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_CHAIN_STEP	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_CREDENTIAL	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_DATABASE_DES TINATION	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_FILE_WATCHER	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_GROUP	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_INCOMPATIBIL ITY	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_JOB	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_JOB_CLASS	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_PROGRAM	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_PROGRAM_ARGU MENT	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_PROGRAM_ARGU MENT	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_RESOURCE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_SCHEDULE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_WINDOW	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	DROP_WINDOW_GROUP	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	ENABLE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	END_DETACHED_JOB_ RUN	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	EVALUATE_RUNNING_ CHAIN	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	GET_AGENT_INFO	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	GET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	GET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	GET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	GET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	GET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	GET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	GET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	GET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	GET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	GET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	GET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	GET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	GET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	GET_FILE	PRAGMA AUTO with COMMIT

Sche ma	Package	Procedure	Pragma
SYS	DBMS_SCHEDULER	GET_FILE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	GET_FILE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	GET_SCHEDULER_ATT RIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	PURGE_LOG	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	PUT_FILE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	PUT_FILE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	PUT_FILE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	REMOVE_EVENT_QUEU E_SUBSCRIBER	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	REMOVE_FROM_INCOM PATIBILITY	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	REMOVE_GROUP_MEMB ER	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	REMOVE_JOB_EMAIL_ NOTIFICATION	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	REMOVE_WINDOW_GRO UP_MEMBER	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	RESET_JOB_ARGUMEN T_VALUE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	RESET_JOB_ARGUMEN T_VALUE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	RUN_CHAIN	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	RUN_CHAIN	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	SET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	SET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	SET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	SET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	SET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	SET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	SET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	SET_ATTRIBUTE_NUL L	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	SET_JOB_ANYDATA_V ALUE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	SET_JOB_ANYDATA_V ALUE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	SET_JOB_ARGUMENT_ VALUE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	SET_JOB_ARGUMENT_ VALUE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	SET_JOB_ATTRIBUTE S	PRAGMA AUTO with COMMIT

Sche ma	Package	Procedure	Pragma
SYS	DBMS_SCHEDULER	SET_RESOURCE_CONS TRAINT	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	SET_SCHEDULER_ATT RIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SCHEDULER	SHOW_ERRORS	PRAGMA AUTO with COMMIT
SYS	DBMS_SQL_TRANSL ATOR	CLEAR_SQL_TRANSLA TION_ERROR	PRAGMA AUTO with COMMIT
SYS	DBMS_SQL_TRANSL ATOR	CREATE_PROFILE	PRAGMA AUTO with COMMIT
SYS	DBMS_SQL_TRANSL ATOR	DEREGISTER_ERROR_ TRANSLATION	PRAGMA AUTO with COMMIT
SYS	DBMS_SQL_TRANSL ATOR	DEREGISTER_SQL_TR ANSLATION	PRAGMA AUTO with COMMIT
SYS	DBMS_SQL_TRANSL ATOR	DROP_PROFILE	PRAGMA AUTO with COMMIT
SYS	DBMS_SQL_TRANSL ATOR	ENABLE_ERROR_TRAN SLATION	PRAGMA AUTO with COMMIT
SYS	DBMS_SQL_TRANSL ATOR	ENABLE_SQL_TRANSL ATION	PRAGMA AUTO with COMMIT
SYS	DBMS_SQL_TRANSL ATOR	REGISTER_ERROR_TR ANSLATION	PRAGMA AUTO with COMMIT
SYS	DBMS_SQL_TRANSL ATOR	REGISTER_SQL_TRAN SLATION	PRAGMA AUTO with COMMIT
SYS	DBMS_SQL_TRANSL ATOR	SET_ATTRIBUTE	PRAGMA AUTO with COMMIT
SYS	DBMS_SQL_TRANSL ATOR	SET_ERROR_TRANSLA TION_COMMENT	PRAGMA AUTO with COMMIT
SYS	DBMS_SQL_TRANSL ATOR	SET_SQL_TRANSLATI ON_COMMENT	PRAGMA AUTO with COMMIT
SYS	DBMS_SQL_TRANSL ATOR	SET_SQL_TRANSLATI ON_MODULE	PRAGMA AUTO with COMMIT
SYS	DBMS_XDS	ALTER_STATIC_ACL_ REFRESH	PRAGMA AUTO
SYS	DBMS_XDS	DISABLE_OLAP_POLI CY	PRAGMA AUTO
SYS	DBMS_XDS	DISABLE_XDS	PRAGMA AUTO
SYS	DBMS_XDS	DROP_OLAP_POLICY	PRAGMA AUTO
SYS	DBMS_XDS	DROP_XDS	PRAGMA AUTO
SYS	DBMS_XDS	ENABLE_OLAP_POLIC Y	PRAGMA AUTO
SYS	DBMS_XDS	ENABLE_XDS	PRAGMA AUTO
SYS	DBMS_XDS	PURGE_ACL_REFRESH _HISTORY	PRAGMA AUTO
SYS	DBMS_XDS	SCHEDULE_STATIC_A CL_REFRESH	PRAGMA AUTO

Sche ma	Package	Procedure	Pragma
SYS	DBMS_XDS	SET_TRACE_LEVEL	PRAGMA AUTO
SYS	DBMS_XDS	XDS\$REFRESH_STATI C_ACL	PRAGMA AUTO
SYS	LOGSTDBY_INTERN AL	EDS_EVOLVE_TABLE_ I	PRAGMA AUTO with COMMIT
SYS	LOGSTDBY_INTERN AL	EDS_REMOVE_TABLE_ I	PRAGMA AUTO with COMMIT
SYS	XS_ACL	ADD_ACL_PARAMETER	PRAGMA AUTO
SYS	XS_ACL	ADD_ACL_PARAMETER	PRAGMA AUTO
SYS	XS_ACL	APPEND_ACES	PRAGMA AUTO
SYS	XS_ACL	APPEND_ACES	PRAGMA AUTO
SYS	XS_ACL	CREATE_ACL	PRAGMA AUTO
SYS	XS_ACL	DELETE_ACL	PRAGMA AUTO
SYS	XS_ACL	REMOVE_ACES	PRAGMA AUTO
SYS	XS_ACL	REMOVE_ACL_PARAME TERS	PRAGMA AUTO
SYS	XS_ACL	REMOVE_ACL_PARAME TERS	PRAGMA AUTO
SYS	XS_ACL	REMOVE_ACL_PARAME TERS	PRAGMA AUTO
SYS	XS_ACL	SET_DESCRIPTION	PRAGMA AUTO
SYS	XS_ACL	SET_PARENT_ACL	PRAGMA AUTO
SYS	XS_ACL	SET_SECURITY_CLAS S	PRAGMA AUTO
SYS	XS_ADMIN_UTIL	GRANT_SYSTEM_PRIV ILEGE	PRAGMA AUTO
SYS	XS_ADMIN_UTIL	REVOKE_SYSTEM_PRI VILEGE	PRAGMA AUTO
SYS	XS_DATA_SECURITY	ADD_COLUMN_CONSTR AINTS	PRAGMA AUTO
SYS	XS_DATA_SECURITY	ADD_COLUMN_CONSTR AINTS	PRAGMA AUTO
SYS	XS_DATA_SECURITY	APPEND_REALM_CONS TRAINTS	PRAGMA AUTO
SYS	XS_DATA_SECURITY	APPEND_REALM_CONS TRAINTS	PRAGMA AUTO
SYS	XS_DATA_SECURITY	APPLY_OBJECT_POLI CY	PRAGMA AUTO
SYS	XS_DATA_SECURITY	CREATE_ACL_PARAME TER	PRAGMA AUTO
SYS	XS_DATA_SECURITY	CREATE_POLICY	PRAGMA AUTO
SYS	XS_DATA_SECURITY	DELETE_ACL_PARAME TER	PRAGMA AUTO

Sche ma	Package	Procedure	Pragma
SYS	XS_DATA_SECURITY	DELETE_POLICY	PRAGMA AUTO
SYS	XS_DATA_SECURITY	DISABLE_OBJECT_POLICY	PRAGMA AUTO
SYS	XS_DATA_SECURITY	ENABLE_OBJECT_POLICY	PRAGMA AUTO
SYS	XS_DATA_SECURITY	REMOVE_COLUMN_CONSTRAINTS	PRAGMA AUTO
SYS	XS_DATA_SECURITY	REMOVE_OBJECT_POLICY	PRAGMA AUTO
SYS	XS_DATA_SECURITY	REMOVE_REALM_CONSTRAINTS	PRAGMA AUTO
SYS	XS_DATA_SECURITY	SET_DESCRIPTION	PRAGMA AUTO
SYS	XS_NAMESPACE	ADD_ATTRIBUTES	PRAGMA AUTO
SYS	XS_DATA_SECURITY	ADD_ATTRIBUTES	PRAGMA AUTO
SYS	XS_DATA_SECURITY	CREATE_TEMPLATE	PRAGMA AUTO
SYS	XS_DATA_SECURITY	DELETE_TEMPLATE	PRAGMA AUTO
SYS	XS_DATA_SECURITY	REMOVE_ATTRIBUTES	PRAGMA AUTO
SYS	XS_DATA_SECURITY	REMOVE_ATTRIBUTES	PRAGMA AUTO
SYS	XS_DATA_SECURITY	REMOVE_ATTRIBUTES	PRAGMA AUTO
SYS	XS_DATA_SECURITY	SET_DESCRIPTION	PRAGMA AUTO
SYS	XS_DATA_SECURITY	SET_HANDLER	PRAGMA AUTO
SYS	XS_PRINCIPAL	ADD_PROXY_TO_DBUSER	PRAGMA AUTO
SYS	XS_PRINCIPAL	ADD_PROXY_USER	PRAGMA AUTO
SYS	XS_PRINCIPAL	ADD_PROXY_USER	PRAGMA AUTO
SYS	XS_PRINCIPAL	CREATE_DYNAMIC_ROLE	PRAGMA AUTO
SYS	XS_PRINCIPAL	CREATE_ROLE	PRAGMA AUTO
SYS	XS_PRINCIPAL	CREATE_USER	PRAGMA AUTO
SYS	XS_PRINCIPAL	DELETE_PRINCIPAL	PRAGMA AUTO
SYS	XS_PRINCIPAL	ENABLE_BY_DEFAULT	PRAGMA AUTO
SYS	XS_PRINCIPAL	ENABLE_ROLES_BY_DEFAULT	PRAGMA AUTO
SYS	XS_PRINCIPAL	GRANT_ROLES	PRAGMA AUTO
SYS	XS_PRINCIPAL	GRANT_ROLES	PRAGMA AUTO

Sche ma	Package	Procedure	Pragma
SYS	XS_PRINCIPAL	REMOVE_PROXY_FROM_DBUSER	PRAGMA AUTO
SYS	XS_PRINCIPAL	REMOVE_PROXY_USERS	PRAGMA AUTO
SYS	XS_PRINCIPAL	REMOVE_PROXY_USERS	PRAGMA AUTO
SYS	XS_PRINCIPAL	REVOKE_ROLES	PRAGMA AUTO
SYS	XS_PRINCIPAL	REVOKE_ROLES	PRAGMA AUTO
SYS	XS_PRINCIPAL	REVOKE_ROLES	PRAGMA AUTO
SYS	XS_PRINCIPAL	SET_ACL	PRAGMA AUTO
SYS	XS_PRINCIPAL	SET_DESCRIPTION	PRAGMA AUTO
SYS	XS_PRINCIPAL	SET_DYNAMIC_ROLE_DURATION	PRAGMA AUTO
SYS	XS_PRINCIPAL	SET_DYNAMIC_ROLE_SCOPE	PRAGMA AUTO
SYS	XS_PRINCIPAL	SET_EFFECTIVE_DATES	PRAGMA AUTO
SYS	XS_PRINCIPAL	SET_GUID	PRAGMA AUTO
SYS	XS_PRINCIPAL	SET_PROFILE	PRAGMA AUTO
SYS	XS_PRINCIPAL	SET_USER_SCHEMA	PRAGMA AUTO
SYS	XS_PRINCIPAL	SET_USER_STATUS	PRAGMA AUTO
SYS	XS_PRINCIPAL_IN_T	SET_VERIFIER_HELPER	PRAGMA AUTO
SYS	XS_ROLESET	ADD_ROLES	PRAGMA AUTO
SYS	XS_ROLESET	ADD_ROLES	PRAGMA AUTO
SYS	XS_ROLESET	CREATE_ROLESET	PRAGMA AUTO
SYS	XS_ROLESET	DELETE_ROLESET	PRAGMA AUTO
SYS	XS_ROLESET	REMOVE_ROLES	PRAGMA AUTO
SYS	XS_ROLESET	REMOVE_ROLES	PRAGMA AUTO
SYS	XS_ROLESET	REMOVE_ROLES	PRAGMA AUTO
SYS	XS_ROLESET	SET_DESCRIPTION	PRAGMA AUTO
SYS	XS_SECURITY_CLASSES	ADD_IMPLIED_PRIVILEGES	PRAGMA AUTO
SYS	XS_SECURITY_CLASSES	ADD_IMPLIED_PRIVILEGES	PRAGMA AUTO
SYS	XS_SECURITY_CLASSES	ADD_PARENTS	PRAGMA AUTO
SYS	XS_SECURITY_CLASSES	ADD_PARENTS	PRAGMA AUTO
SYS	XS_SECURITY_CLASSES	ADD_PRIVILEGES	PRAGMA AUTO
SYS	XS_SECURITY_CLASSES	ADD_PRIVILEGES	PRAGMA AUTO

Sche ma	Package	Procedure	Pragma
SYS	XS_SECURITY_CLA SS	CREATE_SECURITY_C LASS	PRAGMA AUTO
SYS	XS_SECURITY_CLA SS	DELETE_SECURITY_C LASS	PRAGMA AUTO
SYS	XS_SECURITY_CLA SS	REMOVE_IMPLIED_PR IVILEGES	PRAGMA AUTO
SYS	XS_SECURITY_CLA SS	REMOVE_IMPLIED_PR IVILEGES	PRAGMA AUTO
SYS	XS_SECURITY_CLA SS	REMOVE_IMPLIED_PR IVILEGES	PRAGMA AUTO
SYS	XS_SECURITY_CLA SS	REMOVE_PARENTS	PRAGMA AUTO
SYS	XS_SECURITY_CLA SS	REMOVE_PARENTS	PRAGMA AUTO
SYS	XS_SECURITY_CLA SS	REMOVE_PARENTS	PRAGMA AUTO
SYS	XS_SECURITY_CLA SS	REMOVE_PRIVILEGES	PRAGMA AUTO
SYS	XS_SECURITY_CLA SS	REMOVE_PRIVILEGES	PRAGMA AUTO
SYS	XS_SECURITY_CLA SS	REMOVE_PRIVILEGES	PRAGMA AUTO
SYS	XS_SECURITY_CLA SS	SET_DESCRIPTION	PRAGMA AUTO
SYS	DBMS_RESCONFIG	ADDREPOSITORYRESC ONFIG	PRAGMA AUTO with COMMIT
SYS	DBMS_RESCONFIG	ADDRESCONFIG	PRAGMA AUTO
SYS	DBMS_RESCONFIG	APPENDRESCONFIG	PRAGMA AUTO
SYS	DBMS_RESCONFIG	DELETEREPOSITORYR ESCONFIG	PRAGMA AUTO with COMMIT
SYS	DBMS_RESCONFIG	DELETERESCONFIG	PRAGMA AUTO
SYS	DBMS_RESCONFIG	DELETERESCONFIG	PRAGMA AUTO
SYS	DBMS_XDBZ	DISABLE_HIERARCHY	PRAGMA AUTO with COMMIT
SYS	DBMS_XDBZ	ENABLE_HIERARCHY	PRAGMA AUTO with COMMIT
SYS	DBMS_XDB_VERSIO N	CHECKIN_INT	PRAGMA AUTO
SYS	DBMS_XDB_VERSIO N	CHECKOUT	PRAGMA AUTO
SYS	DBMS_XDB_VERSIO N	MAKEVERSIONED_INT	PRAGMA AUTO
SYS	DBMS_XDB_VERSIO N	UNCHECKOUT_INT	PRAGMA AUTO
SYS	DBMS_XLSB	DELETERESOURCE	PRAGMA AUTO
SYS	DBMS_XLSB	DELNAMELOCKS	PRAGMA AUTO
SYS	DBMS_XLSB	INSERTRESOURCE	PRAGMA AUTO

Sche ma	Package	Procedure	Pragma
SYS	DBMS_XLSB	INSERTRESOURCENXO B	PRAGMA AUTO
SYS	DBMS_XLSB	INSERTRESOURCENXO BCLOB	PRAGMA AUTO
SYS	DBMS_XLSB	INSERTRESOURCEREF	PRAGMA AUTO
SYS	DBMS_XLSB	INSERTTOHTABLE	PRAGMA AUTO
SYS	DBMS_XLSB	INSERTTOUSERHTAB	PRAGMA AUTO
SYS	DBMS_XLSB	LINKRESOURCE	PRAGMA AUTO
SYS	DBMS_XLSB	SAVEACL	PRAGMA AUTO
SYS	DBMS_XLSB	SETREFCOUNT	PRAGMA AUTO
SYS	DBMS_XLSB	TOUCHOID	PRAGMA AUTO

PL/SQL Procedures with Pragma MANUAL

For the procedures and packages pragma-ed MANUAL, the top-level PL/SQL API is not called.

Schem a	Package	Procedure	Pragma
SYS	DBMS_AQ	AQ\$_BACKGROUND_OPE R_PAS	PRAGMA MANUAL
SYS	DBMS_AQ	DEQUEUE_INTERNAL_P AS	PRAGMA MANUAL
SYS	DBMS_AQ	ENQUEUE_INT_UNSHAR DED_PAS	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	ALTER_PROPAGATION_ SCHEDULE_INT	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	ALTER_QUEUE_INT	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	ALTER_QUEUE_TABLE_ INT	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	ALTER_SUBSCRIBER_1 1G_INT	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	CREATE_QUEUE_INT	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	CREATE_QUEUE_TABLE _INT	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	DISABLE_PROP_SCHED ULE_INT	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	DROP_QUEUE_INT	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	DROP_QUEUE_TABLE_I NT	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	ENABLE_PROP_SCHEDU LE_INT	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	GRANT_QUEUE_PRIVIL EGE_INT	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	MIGRATE_QUEUE_TABL E_INT	PRAGMA MANUAL

Schem a	Package	Procedure	Pragma
SYS	DBMS_AQADM_SYS	PURGE_QUEUE_TABLE	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	RECOVER_PROPAGATIO N_INT	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	REMOVE_ORPHMSGS_NR	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	REMOVE_SUBSCRIBER_ 11G	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	REVOKE_QUEUE_PRIVI LEGE_INT	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	SCHEDULE_PROPAGATI ON_INT	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	START_QUEUE_INT	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	STOP_QUEUE_INT	PRAGMA MANUAL
SYS	DBMS_AQADM_SYS	UNSCHEDULE_PROPAGA TION_INT	PRAGMA MANUAL
SYS	DBMS_GOLDENGATE_ AUTH	GRANT_ADMIN_PRIVIL EGE	PRAGMA MANUAL with COMMIT
SYS	DBMS_GOLDENGATE_ AUTH	REVOKE_ADMIN_PRIVI LEGE	PRAGMA MANUAL with COMMIT
SYS	DBMS_INTERNAL_LO GSTDBY	EDS_EVOLVE_TABLE_S TART	PRAGMA MANUAL with COMMIT
SYS	DBMS_PRVTAQIS	SUBID_REPLICATE_IN T	PRAGMA MANUAL
SYS	LOGSTDBY_INTERNA L	EDS_ADD_TABLE_I	PRAGMA MANUAL with COMMIT
SYS	XS_ADMIN_UTIL	DROP_SCHEMA_OBJECT S	PRAGMA MANUAL
XDB	DBMS_XDBZ0	DISABLE_HIERARCHY_ INTERNAL	PRAGMA MANUAL
XDB	DBMS_XDBZ0	ENABLE_HIERARCHY_I NTERNAL	PRAGMA MANUAL

PL/SQL Procedures with Pragma NONE

For the procedures and packages pragma-ed NONE, PL/SQL markers are not generated and no grouping is performed. Redo logs generated by these procedures are applied or skipped based on table level replication semantics.

Sche ma	Package	Procedure	Pragma
DVSY	DBMS_MACADM	DISABLE_EVENT	PRAGMA NONE
DVSY	DBMS_MACADM	DV_SANITY_CHECK	PRAGMA NONE
DVSY	DBMS_MACADM	ENABLE_EVENT	PRAGMA NONE
DVSY	DBMS_MACADM	SET_PRESERVE_CASE	PRAGMA NONE
DVSY	DBMS_MACADM	INIT_SESSION	PRAGMA NONE

Sche ma	Package	Procedure	Pragma
DVSY	DBMS_MACADM	UPDATE_POLICY_LAB EL_CONTEXT	PRAGMA NONE
DVSY	DBMS_MACOLS_SE SION	LABEL_AUDIT_RAISE	PRAGMA NONE
DVSY	DBMS_MACOLS_SE SION	RESTORE_DEFAULT_L ABELS	PRAGMA NONE
DVSY	DBMS_MACOLS_SE SION	SET_POLICY_LABEL_ CONTEXT	PRAGMA NONE
DVSY	DBMS_MACOUT	DISABLE	PRAGMA NONE
DVSY	DBMS_MACOUT	ENABLE	PRAGMA NONE
DVSY	DBMS_MACOUT	PL	PRAGMA NONE
DVSY	DBMS_MACOUT	PUT_LINE	PRAGMA NONE
DVSY	DBMS_MACOUT	SET_FACTOR	PRAGMA NONE
DVSY	DBMS_MACSEC_RO LES	SET_ROLE	PRAGMA NONE
DVSY	DBMS_MACSEC_RO LES	EVALUATE	PRAGMA NONE
DVSY	DBMS_MACSEC_RO LES	EVALUATE_TR	PRAGMA NONE
DVSY	DBMS_MACSEC_RO LES	EVALUATE_WR	PRAGMA NONE
DVSY	DBMS_MACUTL	CHECK_DVSY_DML_A LLOWED	PRAGMA NONE
DVSY	DBMS_MACUTL	RAISE_ERROR	PRAGMA NONE
DVSY	DBMS_MACUTL	RAISE_UNAUTHORIZE D_OPERATION	PRAGMA NONE
DVSY	EVENT	SET	PRAGMA NONE
DVSY	EVENT	SETDEFAULT	PRAGMA NONE
DVSY	EVENT	SET_C	PRAGMA NONE
SYS	DBMS_AQ	AQ\$_DEQUEUE	PRAGMA NONE
SYS	DBMS_AQ	AQ\$_DEQUEUE	PRAGMA NONE
SYS	DBMS_AQ	AQ\$_DEQUEUE	PRAGMA NONE
SYS	DBMS_AQ	AQ\$_DEQUEUE	PRAGMA NONE
SYS	DBMS_AQ	BIND_AGENT	PRAGMA NONE
SYS	DBMS_AQ	DEQUEUE	PRAGMA NONE
SYS	DBMS_AQ	DEQUEUE	PRAGMA NONE
SYS	DBMS_AQ	DEQUEUE	PRAGMA NONE
SYS	DBMS_AQ	ENQUEUE	PRAGMA NONE
SYS	DBMS_AQ	ENQUEUE	PRAGMA NONE
SYS	DBMS_AQ	ENQUEUE	PRAGMA NONE
SYS	DBMS_AQ	LISTEN	PRAGMA NONE
SYS	DBMS_AQ	LISTEN	PRAGMA NONE
SYS	DBMS_AQ	POST	PRAGMA NONE

Sche ma	Package	Procedure	Pragma
SYS	DBMS_AQ	REGISTER	PRAGMA NONE
SYS	DBMS_AQ	UNBIND_AGENT	PRAGMA NONE
SYS	DBMS_AQ	UNREGISTER	PRAGMA NONE
SYS	DBMS_AQADM	ADD_ALIAS_TO_LDAP	PRAGMA NONE
SYS	DBMS_AQADM	ADD_CONNECTION_TO _LDAP	PRAGMA NONE
SYS	DBMS_AQADM	ADD_CONNECTION_TO _LDAP	PRAGMA NONE
SYS	DBMS_AQADM	ADD_SUBSCRIBER	PRAGMA NONE
SYS	DBMS_AQADM	ALTER_PROPAGATION _SCHEDULE	PRAGMA NONE
SYS	DBMS_AQADM	ALTER_QUEUE	PRAGMA NONE
SYS	DBMS_AQADM	ALTER_QUEUE_TABLE	PRAGMA NONE
SYS	DBMS_AQADM	ALTER_SHARDED_QUE UE	PRAGMA NONE
SYS	DBMS_AQADM	ALTER_SUBSCRIBER	PRAGMA NONE
SYS	DBMS_AQADM	ALTER_SUBSCRIBER	PRAGMA NONE
SYS	DBMS_AQADM	CREATE_EXCEPTION_ QUEUE	PRAGMA NONE
SYS	DBMS_AQADM	CREATE_NP_QUEUE	PRAGMA NONE
SYS	DBMS_AQADM	CREATE_QUEUE	PRAGMA NONE
SYS	DBMS_AQADM	CREATE_QUEUE_TABL E	PRAGMA NONE
SYS	DBMS_AQADM	CREATE_SHARDED_QU EUE	PRAGMA NONE
SYS	DBMS_AQADM	DEL_ALIAS_FROM_LD AP	PRAGMA NONE
SYS	DBMS_AQADM	DEL_CONNECTION_FR OM_LDAP	PRAGMA NONE
SYS	DBMS_AQADM	DISABLE_PROPAGATI ON_SCHEDULE	PRAGMA NONE
SYS	DBMS_AQADM	DROP_QUEUE	PRAGMA NONE
SYS	DBMS_AQADM	DROP_QUEUE_TABLE	PRAGMA NONE
SYS	DBMS_AQADM	DROP_SHARDED_QUEU E	PRAGMA NONE
SYS	DBMS_AQADM	ENABLE_JMS_TYPES	PRAGMA NONE
SYS	DBMS_AQADM	ENABLE_PROPAGATIO N_SCHEDULE	PRAGMA NONE
SYS	DBMS_AQADM	GET_PROP_SEQNO	PRAGMA NONE
SYS	DBMS_AQADM	GET_REPLAY_INFO	PRAGMA NONE
SYS	DBMS_AQADM	GET_TYPE_INFO	PRAGMA NONE
SYS	DBMS_AQADM	GET_TYPE_INFO	PRAGMA NONE
SYS	DBMS_AQADM	GET_WATERMARK	PRAGMA NONE

Sche ma	Package	Procedure	Pragma
SYS	DBMS_AQADM	GRANT_QUEUE_PRIVILEGE	PRAGMA NONE
SYS	DBMS_AQADM	MIGRATE_QUEUE_TABLE	PRAGMA NONE
SYS	DBMS_AQADM	NONREPUDIATE_RECEIVER	PRAGMA NONE
SYS	DBMS_AQADM	NONREPUDIATE_RECEIVER	PRAGMA NONE
SYS	DBMS_AQADM	NONREPUDIATE_SENDER	PRAGMA NONE
SYS	DBMS_AQADM	NONREPUDIATE_SENDER	PRAGMA NONE
SYS	DBMS_AQADM	PURGE_QUEUE_TABLE	PRAGMA NONE
SYS	DBMS_AQADM	RECOVER_PROPAGATION	PRAGMA NONE
SYS	DBMS_AQADM	REMOVE_SUBSCRIBER	PRAGMA NONE
SYS	DBMS_AQADM	RESET_REPLAY_INFO	PRAGMA NONE
SYS	DBMS_AQADM	REVOKE_QUEUE_PRIVILEGE	PRAGMA NONE
SYS	DBMS_AQADM	SCHEDULE_PROPAGATION	PRAGMA NONE
SYS	DBMS_AQADM	SET_WATERMARK	PRAGMA NONE
SYS	DBMS_AQADM	START_QUEUE	PRAGMA NONE
SYS	DBMS_AQADM	START_TIME_MANAGER	PRAGMA NONE
SYS	DBMS_AQADM	STOP_QUEUE	PRAGMA NONE
SYS	DBMS_AQADM	STOP_TIME_MANAGER	PRAGMA NONE
SYS	DBMS_AQADM	UNSCHEDULE_PROPAGATION	PRAGMA NONE
SYS	DBMS_AQADM	VERIFY_QUEUE_TYPES	PRAGMA NONE
SYS	DBMS_AQADM	VERIFY_QUEUE_TYPES_GET_NRP	PRAGMA NONE
SYS	DBMS_AQADM	VERIFY_QUEUE_TYPES_NO_QUEUE	PRAGMA NONE
SYS	DBMS_AQELM	GET_MAILHOST	PRAGMA NONE
SYS	DBMS_AQELM	GET_MAILPORT	PRAGMA NONE
SYS	DBMS_AQELM	GET_PROXY	PRAGMA NONE
SYS	DBMS_AQELM	GET_SENDFROM	PRAGMA NONE
SYS	DBMS_AQELM	GET_TXTIMEOUT	PRAGMA NONE
SYS	DBMS_AQELM	HTTP_SEND	PRAGMA NONE
SYS	DBMS_AQELM	SEND_EMAIL	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$DEQUEUE_IN	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$DEQUEUE_IN	PRAGMA NONE

Sche ma	Package	Procedure	Pragma
SYS	DBMS_AQIN	AQ\$_DEQUEUE_IN	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$_DEQUEUE_IN	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$_DEQUEUE_IN	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$_DEQUEUE_RAW	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$_DEQUEUE_RAW	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$_ENQUEUE_OBJ	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$_ENQUEUE_OBJ	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$_ENQUEUE_OBJ_N O_RECPL	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$_ENQUEUE_OBJ_N O_RECPL	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$_ENQUEUE_RAW	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$_JMS_ENQUEUE_B YTES_MESSAGE	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$_JMS_ENQUEUE_M AP_MESSAGE	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$_JMS_ENQUEUE_O BJECT_MESSAGE	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$_JMS_ENQUEUE_S TREAM_MESSAGE	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$_JMS_ENQUEUE_T EXT_MESSAGE	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$_LISTEN	PRAGMA NONE
SYS	DBMS_AQIN	AQ\$_QUEUE_SUBSCRI BERS	PRAGMA NONE
SYS	DBMS_AQIN	SET_DEQ_SORT	PRAGMA NONE
SYS	DBMS_AQIN	SET_MULTI_RETRY	PRAGMA NONE
SYS	DBMS_AQJMS	AQ\$_GET_PROP_STAT	PRAGMA NONE
SYS	DBMS_AQJMS	AQ\$_GET_TRANS_TYP E	PRAGMA NONE
SYS	DBMS_AQJMS	AQ\$_REGISTER	PRAGMA NONE
SYS	DBMS_AQJMS	AQ\$_UNREGISTER	PRAGMA NONE
SYS	DBMS_AQJMS	AQ\$_UPDATE_PROP_S TAT_QNAME	PRAGMA NONE
SYS	DBMS_AQJMS	CLEAR_DBSESSION_G UID	PRAGMA NONE
SYS	DBMS_AQJMS	CLEAR_GLOBAL_AQCL NTDB_CTX_CLNT	PRAGMA NONE
SYS	DBMS_AQJMS	CLEAR_GLOBAL_AQCL NTDB_CTX_DB	PRAGMA NONE
SYS	DBMS_AQJMS	GET_DB_USERNAME_F OR_AGENT	PRAGMA NONE
SYS	DBMS_AQJMS	SET_DBSESSION_GUI D	PRAGMA NONE

Sche ma	Package	Procedure	Pragma
SYS	DBMS_AQJMS	SET_GLOBAL_AQCLNT DB_CTX	PRAGMA NONE
SYS	DBMS_AQJMS	SUBSCRIBER_EXISTS	PRAGMA NONE
SYS	DBMS_AQJMS	SUBSCRIBER_EXISTS	PRAGMA NONE
SYS	DBMS_ISCHED	GET_AGENT_PASS_VE RIFIER	PRAGMA NONE
SYS	DBMS_ISCHED	OBFUSCATE_CREDENT IAL_PASSWORD	PRAGMA NONE
SYS	DBMS_REDEFINITI ON	CAN_REDEF_TABLE	PRAGMA NONE
SYS	DBMS_REDEFINITI ON	REDEF_TABLE	PRAGMA NONE
SYS	DBMS_SCHEDULER	CHECK_SYS_PRIVS	PRAGMA NONE
SYS	DBMS_SCHEDULER	CLOSE_WINDOW	PRAGMA NONE
SYS	DBMS_SCHEDULER	CREATE_CALENDAR_S TRING	PRAGMA NONE
SYS	DBMS_SCHEDULER	CREATE_CREDENTIAL	PRAGMA NONE
SYS	DBMS_SCHEDULER	EVALUATE_CALENDAR _STRING	PRAGMA NONE
SYS	DBMS_SCHEDULER	FILE_WATCH_FILTER	PRAGMA NONE
SYS	DBMS_SCHEDULER	GENERATE_EVENT_LI ST	PRAGMA NONE
SYS	DBMS_SCHEDULER	GENERATE_JOB_NAME	PRAGMA NONE
SYS	DBMS_SCHEDULER	GET_AGENT_VERSION	PRAGMA NONE
SYS	DBMS_SCHEDULER	GET_CHAIN_RULE_AC TION	PRAGMA NONE
SYS	DBMS_SCHEDULER	GET_CHAIN_RULE_CO NDITION	PRAGMA NONE
SYS	DBMS_SCHEDULER	GET_DEFAULT_VALUE	PRAGMA NONE
SYS	DBMS_SCHEDULER	GET_JOB_STEP_CF	PRAGMA NONE
SYS	DBMS_SCHEDULER	GET_SYS_TIME_ZONE _NAME	PRAGMA NONE
SYS	DBMS_SCHEDULER	GET_VARCHAR2_VALU E	PRAGMA NONE
SYS	DBMS_SCHEDULER	GET_VARCHAR2_VALU E	PRAGMA NONE
SYS	DBMS_SCHEDULER	IS_SCHEDULER_CREA TED_AGENT	PRAGMA NONE
SYS	DBMS_SCHEDULER	OPEN_WINDOW	PRAGMA NONE
SYS	DBMS_SCHEDULER	RESOLVE_CALENDAR_ STRING	PRAGMA NONE
SYS	DBMS_SCHEDULER	RESOLVE_CALENDAR_ STRING	PRAGMA NONE
SYS	DBMS_SCHEDULER	RESOLVE_NAME	PRAGMA NONE

Sche ma	Package	Procedure	Pragma
SYS	DBMS_SCHEDULER	RUN_JOB	PRAGMA NONE
SYS	DBMS_SCHEDULER	SET_AGENT_REGISTR ATION_PASS	PRAGMA NONE
SYS	DBMS_SCHEDULER	STIME	PRAGMA NONE
SYS	DBMS_SCHEDULER	STOP_JOB	PRAGMA NONE
SYS	DBMS_SCHEDULER	SUBMIT_REMOTE_EXT ERNAL_JOB	PRAGMA NONE
SYS	XS_PRINCIPAL	SET_PASSWORD	PRAGMA NONE
SYS	XS_PRINCIPAL	SET_VERIFIER	PRAGMA NONE

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 15-2 Displaying Information About the Packages Supported for Oracle GoldenGate Procedural Replication

This query displays the following information about the packages:

- The owner of each package
- The name of each package
- The name of each procedure
- The minimum database release from which the procedure is supported
- Whether there is an exclusion rule that prevents the procedure from being replicated for some database objects

```
COLUMN OWNER FORMAT A10
COLUMN PACKAGE_NAME FORMAT A15
COLUMN PROCEDURE_NAME FORMAT A15
COLUMN MIN_DB_VERSION FORMAT A14
COLUMN EXCLUSION_RULE_EXISTS FORMAT A14
```

```
SELECT OWNER,
       PACKAGE_NAME,
       PROCEDURE_NAME,
       MIN_DB_VERSION,
```



```
EXCLUSION_RULE_EXISTS
FROM DBA_GG_SUPPORTED_PROCEDURES;
```

Your output looks similar to the following:

OWNER	PACKAGE_NAME	PROCEDURE_NAME	MIN_DB_VERSION	EXCLUSION_RULE
XDB	DBMS_XDB_CONFIG	ADDTRUSTMAPPING	12.2	NO
CTXSYS	CTX_DDL	ALTER_INDEX	12.2	NO
SYS	DBMS_FGA	DROP_POLICY	12.2	NO
SYS	XS_ACL	DELETE_ACL	12.2	NO
.				
.				
.				

Monitoring Oracle GoldenGate Procedural Replication

A set of data dictionary views enable you to monitor Oracle GoldenGate procedural replication.

You can use the following views to monitor Oracle GoldenGate procedural replication:

View	Description
DBA_GG_SUPPORTED_PACKAGES	Provides details about supported packages for Oracle GoldenGate procedural replication. When a package is supported and Oracle GoldenGate procedural replication is on, calls to subprograms in the package are replicated.
DBA_GG_SUPPORTED_PROCEDURES	Provides details about the procedures that are supported for Oracle GoldenGate procedural replication.
DBA_GG_PROC_OBJECT_EXCLUSION	Provides details about all database objects that are on the exclusion list for Oracle GoldenGate procedural replication. A database object is added to the exclusion list using the <code>INSERT_PROCREP_EXCLUSION_OBJ</code> procedure in the <code>DBMS_GOLDENGATE_ADM</code> package. When a database object is on the exclusion list, execution of a subprogram in the package is not replicated if the subprogram operates on the excluded object.

1. Connect to the database as `sys (sqlplus, sqlcl, or sqldeveloper)` not as an Oracle GoldenGate administrator.
2. Query the views related to Oracle GoldenGate procedural replication.

16

Configuring Oracle GoldenGate in a Multitenant Container Database

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

Topics:

- [Capturing from Pluggable Databases](#)
- [Applying to Pluggable Databases](#)
- [Excluding Objects from the Configuration](#)
- [Other Requirements for Multitenant Container Databases](#)
This topic describes the special requirements that apply to replication to and from multitenant container databases.

Capturing from Pluggable Databases

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

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

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

Applying to Pluggable Databases

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

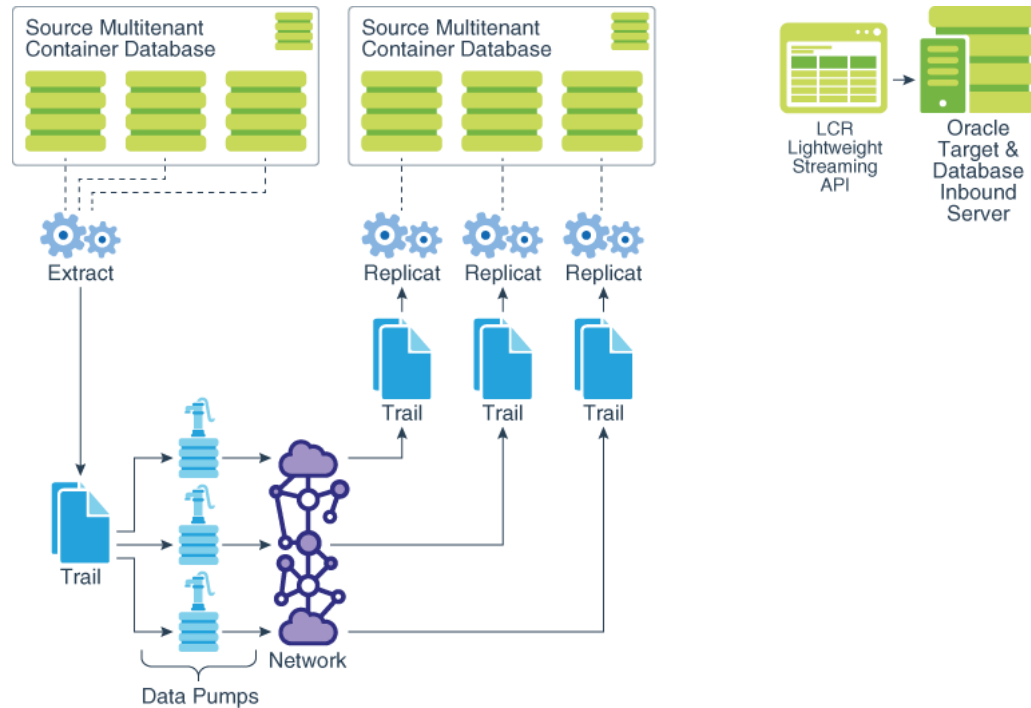
```
SOURCECATALOG pdb1
MAP schema_1.tab, TARGET 1;
MAP schema_1.seq, TARGET 1;
SOURCECATALOG pdb2
```

```
MAP schema_2.tab, TARGET 2;
MAP schema_2.seq, TARGET 2;
```

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

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

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



Excluding Objects from the Configuration

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

Other Requirements for Multitenant Container Databases

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

The requirements are:

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


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

FLUSH SEQUENCE for Multitenant Database

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

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

```
Environment Information OGG 18.1 Oracle 12c to Oracle 12c Replication,
Integrated Extract, Parallel Replicat
Source: CDB GOLD, PDB CERTMISSN
Target: CDB PLAT, PDB CERTDSQ
Source OGG Configuration
  Container User: C##GGADMIN
  PDB User for Sequences: GGATE
sqlplus / as sysdba
SQL> alter session set container=CERTMISSN;
SQL> create user ggate identified by password default tablespace users
temporary tablespace temp quota unlimited on users container=current;
```

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

When prompted enter

```
GGATE GLOBALS  
GGSHEMA GGATE
```

FLUSH SEQUENCE:

```
GGSCI> DBLOGIN USERIDALIAS GGADMIN DOMAIN GOLD_QC_CDB$ROOT  
  
GGSCI> FLUSH SEQUENCE CERTMISSN.SRCSHEMA1.*
```

Target Oracle GoldenGate Configuration:

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

When prompted enter GGATE.

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

Using Oracle GoldenGate with Autonomous Database

You can replicate data to Oracle Autonomous Database using Oracle GoldenGate.

Topics:

- [About Capturing and Replicating Data Using Autonomous Databases](#)
- [Details of Support When Using Oracle GoldenGate with Autonomous Databases](#)
- [Configuring Extract to Capture from an Autonomous Database](#)
- [Configuring Replicat to Apply to an Autonomous Database](#)

About Capturing and Replicating Data Using Autonomous Databases

You can capture changes from the Oracle Autonomous Database instance and replicate to any **target database** or platform that Oracle GoldenGate supports, including another Oracle Autonomous Database instance.

Use Case: When Using Oracle GoldenGate with Autonomous Databases

Using Oracle GoldenGate in the Oracle Autonomous Database can be configured to support the following scenarios:

- **Scalable Active-Active architecture:** Synchronize changes made across two or more databases to scale out workloads, provide increase resilience and near instantaneous failover across multiple data centers or regions.
- **Real-Time Data Warehouse:** Provide continuous, real-time capture and delivery of changed data between Oracle Autonomous Database systems.
- **Big Data Integration:** With Oracle GoldenGate for Big Data you can replicate data from the Oracle Autonomous Database to provide real-time streaming integration to all platforms supported by Big Data targets.
- **Real-Time Streaming Analytics:** Oracle GoldenGate integrates seamlessly with Oracle Stream Analytics to enable users to identify events of interest by executing queries against event streams in real time. It allows creating custom operational dashboards that provide real-time monitoring, transform streaming data, or raise alerts based on stream analysis.
- **Hybrid Replication:** Oracle GoldenGate replicates data from the Oracle Autonomous Database instance back to on-premise or to another cloud database or platform.

The following features are not available with Always Free Autonomous Databases:

- Supplemental logging
- Oracle GoldenGate Extract

See Always Free Autonomous Database for details.

Details of Support When Using Oracle GoldenGate with Autonomous Databases

Review the supported data types and limitations before replicating data to an 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

The following combinations of Replicats are supported in different modes when using Oracle GoldenGate with Oracle Autonomous Database:

- Parallel Replicat in integrated mode is supported for Oracle Autonomous Database.
- Classic and coordinated Replicats in integrated mode are not supported for Oracle Autonomous Database.
- Classic, coordinated, and parallel Replicats in non-integrated mode are supported for Oracle Autonomous Database.

Data Type Limitations for DDL and DML Replication

See the section [Non-Supported Oracle Data Types](#).

Also see Limitation on the Use of Certain 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.

Configuring Replicat to Apply to an Autonomous Database

You can replicate into the Autonomous Database from any source database or platform that is supported by Oracle GoldenGate.

Topics:

- [Prerequisites for Configuring Oracle GoldenGate Replicat to an Autonomous Database](#)
- [Configure Replicat to Apply to an Autonomous Database](#)

Prerequisites for Configuring Oracle GoldenGate Replicat to an Autonomous Database

Learn about the prerequisites for configuring Oracle GoldenGate data replication to Autonomous Databases.

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 environment is provisioned and running.

To deliver data to the Autonomous Database instance using Oracle GoldenGate, perform the following tasks:

- [Configure Oracle GoldenGate Replicat for an Autonomous Database](#)
- [Obtain the Autonomous Database Client Credentials](#)

Configure Oracle GoldenGate Replicat for an Autonomous Database

Learn the steps to configure Oracle GoldenGate Replicat for an Autonomous Database.

Here are the steps to complete the configuration tasks:

 **Note:**

Instructions are based on the assumption that the source environment is already configured.

1. For Oracle GoldenGate on-premises, make sure that Oracle GoldenGate is installed.

Oracle GoldenGate Classic Architecture supports Autonomous Database capture using Marketplace for Oracle Autonomous Database Serverless

2. (Microservices only) Create a deployment for your Oracle GoldenGate environment. This is the deployment where the Replicat that applies data into the Autonomous Database (ADB) will be created. See [How to Create Deployments for steps to add a deployment](#).
3. The Autonomous Database has a pre-existing user created for Oracle GoldenGate on-premise called `ggadmin`. The `ggadmin` user has been granted the required privileges for Replicat to work. This is the user where any objects used for Oracle GoldenGate processing will be stored, like the checkpoint table and heartbeat objects. By default, this user is locked. To unlock the `ggadmin` user, connect to the Oracle Autonomous Database instance as the `ADMIN` user using any SQL client tool. See [Create Users on Autonomous Database with Database Actions](#).
4. Run the `ALTER USER` command to unlock the `ggadmin` user and set the password for it. This will be used in GGSCI or Admin Client for any `DBLOGIN` operations on the Autonomous Database. It will be used in Replicat to allow Oracle GoldenGate to connect to the Autonomous Database and apply data. See [Create Users on Autonomous Database with Database Actions](#).

```
ALTER USER ggadmin IDENTIFIED BY p0$$word ACCOUNT UNLOCK;
```

Obtain the Autonomous Database Client Credentials

Learn how to establish a connection to your Autonomous Database.

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. 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 into 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 Replicat for an Autonomous Database](#).
3. Follow the steps given in [Obtain the Autonomous Database Client Credentials](#).
4. Log into 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 the `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.oraclec
                loud.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 Replicat becomes unresponsive due to a network timeout or a lost connection, 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 or GGSCI to log into 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 [Configuring Oracle GoldenGate Apply](#).

 **Note:**

You can use classic Replicat, coordinated Replicat, and parallel Replicat in non-integrated mode. Parallel Replicat in integrated mode is also supported for Oracle Autonomous Database.

14. You can now start your Replicat and perform data replication to the Autonomous Database.

 **Note:**

Oracle Autonomous Data Warehouse 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 `AUTORESTART` parameter (Classic Architecture) or configure the `AUTORESTART` profile (Microservices Architecture) to avoid having to manually restart a Replicat when it times out.

Configuring Extract to Capture from an Autonomous Database

Oracle Autonomous Database has a tight integration with Oracle GoldenGate. There are a number of differences when setting up Extract for an Autonomous database instance compared to a traditional Oracle Database.

Oracle Autonomous Database security has been enhanced to ensure that Extract is only able to capture changes from the specific tenant it connected to. However, downstream Extract is not supported.

Before You Begin

Before you start the process of capturing data from the Autonomous Database using Oracle GoldenGate you must first:

1. Unlock the pre-created Oracle GoldenGate database user `ggadmin` in the Autonomous Database.
2. Obtain the Autonomous Database client credentials to connect to the database instance.

Topics:

- [Establishing Oracle GoldenGate Credentials](#)

- [Prerequisites for Configuring Oracle GoldenGate Extract to Capture from Autonomous Databases](#)
- [Configure Extract to Capture from an Autonomous Database](#)

Establishing Oracle GoldenGate Credentials

To capture from an Autonomous Database only the `GGADMIN` account is used. The `GGADMIN` account is created inside the database when the Autonomous Database is provisioned. This account is locked. It must be unlocked before it can be used with Oracle GoldenGate. This account is the same account used for both Extracts and Replicats in the Autonomous Database.

Run the `ALTER USER` command to unlock the `ggadmin` user and set the password for it. See [Creating Users with Autonomous Database with Client-Side Tools](#).

This `ALTER USER` command must be run by the `admin` account user for Autonomous Databases.

```
ALTER USER ggadmin IDENTIFIED BY PASSWORD ACCOUNT UNLOCK;
```

Prerequisites for Configuring Oracle GoldenGate Extract to Capture from Autonomous Databases

Prior to configuring and starting the Extract process to capture from the Autonomous Database, make sure that the following requirements are met:

- Oracle Autonomous Database environment is provisioned and running.
- Autonomous Database-level supplemental logging should be enabled by the `ADMIN` or `GGADMIN`.

Configuring Autonomous Database Supplemental Logging for Extract

To add minimal supplemental logging to your Autonomous Database instance, log into the instance as `GGADMIN` or `ADMIN` account and execute the following commands:

```
ALTER PLUGGABLE DATABASE ADD SUPPLEMENTAL LOG DATA;
```

To `DROP` Autonomous Database-level supplemental logging incase you decide to stop capturing from that database instance:

```
ALTER PLUGGABLE DATABASE DROP SUPPLEMENTAL  
LOG DATA;
```

You can verify that the Autonomous Database-level supplemental logging is configured properly by issuing this SQL statement:

```
SELECT MINIMAL FROM dba_supplemental_logging;
```

The output for this statement is:

```
MINIMAL  
-----  
YES
```

The `MINIMAL` column will be `YES` if supplemental logging has been correctly set for this Autonomous Database instance.

Configure Extract to Capture from an Autonomous Database

Following are the steps to configure an Extract to capture from an Oracle Autonomous Database :

1. Install Oracle GoldenGate for your Oracle Autonomous Database instance.
2. (Microservices only) Create a deployment for your Oracle GoldenGate environment. This is the deployment where the Extract that captures data from the Autonomous Database (ADB) will be created. See [How to Create Deployments](#) for steps to add a deployment.
3. Obtain 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 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 into 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 Databases).

```

Sample Connection Stringadwl_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 or GGSCI to log into the Oracle GoldenGate deployment, depending on whether you are using Microservices or Classic Architecture.
6. Create credentials for the Extract database (or a user with same privileges). In this case, GGADMIN is the user and will be used to connect to the Autonomous Database, and

perform commands that require a database connection. It will also be used in the `USERIDALIAS` parameter for the Extract database connection.

```
ALTER CREDENTIALSTORE ADD USER
ggadmin@dbgraph1_low PASSWORD complex_password alias adb_alias
```

7. Connect to the database using `DBLOGIN`. The `DBLOGIN` user should be the `adb_alias` account user.

```
DBLOGIN USERIDALIAS adb_alias
```

8. Configure supplemental logging on the tables, which you want to capture using `ADD TRANDATA` or `ADD SCHEMATRANDATA`. Remember that you are connected directly to the database instance, so there is no need to include the database name in these commands. Here's an example:

```
ADD TRANDATA HR.EMP
```

or

```
ADD SCHEMATRANDATA HR
```

See [Prerequisites for Configuring Oracle GoldenGate Extract to Capture from Autonomous Databases](#).

9. Add heartbeat table.

```
ADD HEARTBEATTABLE
```

10. Add and configure an Extract to capture from the Oracle Autonomous Database instance. See [Configuring the Primary Extract in Integrated Capture Mode](#) 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 is 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.

A

Optional Parameters for Integrated Modes

This appendix contains optional parameters that may be required when operating Extract in integrated capture mode or Replicat in integrated Replicat mode.

Topics:

- [Additional Parameter Options for Integrated Capture](#)
This section contains additional parameters that may be required for your Extract configuration.
- [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 in GGSCI.

Additional Parameter Options for Integrated Capture

This section contains additional parameters that may be required for your Extract configuration.

Integrated capture 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 [Configuring a Downstream Mining Database](#).
- `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](#).

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 in GGSCI.

The default Replicat configuration as directed in [Configuring Oracle GoldenGate Apply](#) 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 *Reference for Oracle GoldenGate* for more information about the `DBOPTIONS` parameter.

- **COMMIT_SERIALIZATION**: Controls the order in which applied transactions are committed and has 2 modes, `DEPENDENT_TRANSACTIONS` and `FULL`. The default mode for Oracle GoldenGate is `DEPENDENT_TRANSACTIONS` where dependent transactions are applied in the correct order though may not necessarily be applied in source commit order. In `FULL` mode, the source commit order is enforced when applying transactions.
- **BATCHSQL_MODE**: Controls the batch execution scheduling mode including pending dependencies. A pending dependency is a dependency on another transaction that has already been scheduled, but not completely executed. The default is `DEPENDENT`. You can use following three modes:

DEPENDENT

Dependency aware scheduling without an early start. Batched transactions are scheduled when there are no pending dependencies.

DEPENDENT_EAGER

Dependency aware batching with early start. Batched transactions are scheduled irrespective of pending dependencies.

SEQUENTIAL

Sequential batching. Transactions are batched by grouping the transactions sequentially based on the original commit order.

- **DISABLE_ON_ERROR:** Determines whether the apply server is disabled or continues on an unresolved error. The default for Oracle GoldenGate is `N` (continue on errors), however, you can set the option to `Y` if you need to disable the apply server when an error occurs.
- **EAGER_SIZE:** Sets a threshold for the size of a transaction (in number of LCRs) after which Oracle GoldenGate starts applying data before the commit record is received. The default for Oracle GoldenGate is `15100`.
- **ENABLE_XSTREAM_TABLE_STATS:** Controls whether statistics on applied transactions are recorded in the `V$GOLDENGATE_TABLE_STATS` view or not collected at all. The default for Oracle GoldenGate is `Y` (collect statistics).
- **MAX_PARALLELISM:** Limits the number of apply servers that can be used when the load is heavy. This number is reduced again when the workload subsides. The automatic tuning of the number of apply servers is effective only if `PARALLELISM` is greater than 1 and `MAX_PARALLELISM` is greater than `PARALLELISM`. If `PARALLELISM` is equal to `MAX_PARALLELISM`, the number of apply servers remains constant during the workload. The default for Oracle GoldenGate is `50`.
- **MAX_SGA_SIZE:** Controls the amount of shared memory used by the inbound server. The shared memory is obtained from the streams pool of the SGA. The default for Oracle GoldenGate is `INFINITE`.
- **MESSAGE_TRACKING_FREQUENCY:** Controls how often LCRs are marked for high-level LCR tracing through the apply processing. The default value is `2000000`, meaning that every 2 millionth LCR is traced. A value of zero (0) disables LCR tracing.
- **PARALLELISM:** Sets a minimum number of apply servers that can be used under normal conditions. Setting `PARALLELISM` to 1 disables apply parallelism, and transactions are applied with a single apply server process. The default for Oracle GoldenGate is `4`. For Oracle Standard Edition, this must be set to `1`.
- **PARALLELISM_INTERVAL:** Sets the interval in seconds at which the current workload activity is computed. Replicat calculates the mean throughput every `5 X PARALLELISM_INTERVAL` seconds. After each calculation, the apply component can increase or decrease the number of apply servers to try to improve throughput. If throughput is improved, the apply component keeps the new number of apply servers. The parallelism interval is used only if `PARALLELISM` is set to a value greater than one and the `MAX_PARALLELISM` value is greater than the `PARALLELISM` value. The default is `5` seconds.
- **PRESERVE_ENCRYPTION:** Controls whether to preserve encryption for columns encrypted using Transparent Data Encryption. The default for Oracle GoldenGate is `N` (do not apply the data in encrypted form).
- **OPTIMIZE_PROGRESS_TABLE:** Integrated Delivery uses this table to track the transactions that have been applied. It is used for duplicate avoidance in the event of failure or restart. If it is set to `N` (the default), then the progress table is updated synchronously with the apply of each replicated transaction. When set to `Y`, rather than populating the progress table synchronously, markers are dropped into the redo stream so when the apply process starts up, it mines the redo logs for these markers, and then updates the progress table for the previously applied transactions.

- `TRACE_LEVEL`: Controls the level of tracing for the Replicat inbound server. For use only with guidance from Oracle Support. The default for Oracle GoldenGate is 0 (no tracing).
- `WRITE_ALERT_LOG`: Controls whether the Replicat inbound server writes messages to the Oracle alert log. The default for Oracle GoldenGate is Y (yes).

B

Configuring a Downstream Mining Database

This appendix contains instructions for preparing a downstream Oracle mining database to support Extract in integrated capture mode.

For more information about integrated capture, see [Deciding Which Capture Method to Use](#).

For examples of the downstream mining configuration, see [Example Downstream Mining Configuration](#).

Topics:

- [Evaluating Capture Options for a Downstream Deployment](#)
Downstream deployment allows you to offload the source database.
- [Preparing the Source Database for Downstream Deployment](#)
The source database ships its redo logs to a downstream database, and Extract uses the logmining server at the downstream database to mine the redo logs.
- [Preparing the Downstream Mining Database](#)
A downstream mining database can accept both archived logs and online redo logs from a source database.

Evaluating Capture Options for a Downstream Deployment

Downstream deployment allows you to offload the source database.

A downstream mining database can accept both archived logs and online redo logs from a source database.

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 bitsize, 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 bitsize.

Preparing the Source Database for Downstream Deployment

The source database ships its redo logs to a downstream database, and Extract uses the logmining server at the downstream database to mine the redo logs.

This section guides you in the process of:

- [Creating the Source User Account](#)
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.
- [Configuring Redo Transport from Source 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.

Creating the Source User Account

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.

The source user is specified by the `USERIDALIAS` parameter.

To assign the required privileges, follow the procedure in [Establishing Oracle GoldenGate Credentials](#)

Configuring Redo Transport from Source 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*. From Oracle Database 12.2.0.1 onward, the archived logs sent to downstream are purged automatically in downstream database as long as it is stored on Flash Recovery Area (FRA).

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 Oracle Net so that each source database can communicate with the mining database. For instructions, see *Oracle Database Net Services Administrator's Guide*.
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. For more information about authentication requirements for redo transport, see *Preparing the Primary Database for Standby Database Creation* in *Oracle Data Guard Concepts and Administration*.
3. At each source database, configure one `LOG_ARCHIVE_DEST_n` initialization parameter to transmit redo data to the downstream mining database. Set the attributes of this parameter as shown in one of the following examples, depending on whether real-time or archived-log-only capture mode is to be used.

- Example for real-time capture at the downstream logmining server, where the source database sends its online redo logs to the downstream database:

```
ALTER SYSTEM
SET LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC NOREGISTER
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) DB_UNIQUE_NAME=dbmscap'
```

- Example for archived-log-only capture at the downstream logmining server:

```
ALTER SYSTEM SET
LOG_ARCHIVE_DEST_2='SERVICE=DBMSCAP.EXAMPLE.COM ASYNC NOREGISTER
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
TEMPLATE=/usr/oracle/log_for_dbms1/dbms1_arch_%t_%s_%r.log
DB_UNIQUE_NAME=dbmscap'
```

 **Note:**

When using an archived-log-only downstream mining database, you must specify a value for the `TEMPLATE` attribute. Oracle also recommends that you use the `TEMPLATE` clause in the source databases so that the log files from all remote source databases are kept separated from the local database log files, and from each other.

4. At the source database, set a value of `ENABLE` for the `LOG_ARCHIVE_DEST_STATE_n` initialization parameter that corresponds with the `LOG_ARCHIVE_DEST_n` parameter that

corresponds to the destination for the downstream mining database, as shown in the following example.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

5. At the source database, and at the downstream mining database, set the `DG_CONFIG` attribute of the `LOG_ARCHIVE_CONFIG` initialization parameter to include the `DB_UNIQUE_NAME` of the source database and the downstream database, as shown in the following example.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1,dbmscap)'
```

Preparing the Downstream Mining Database

A downstream mining database can accept both archived logs and online redo logs from a source database.

The following sections explain how to prepare the downstream mining database:

- [Creating the Downstream Mining User Account](#)
- [Configuring the Mining Database to Archive Local Redo Log Files](#)
- [Preparing a Downstream Mining Database for Real-time Capture](#)

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 [Establishing Oracle GoldenGate Credentials](#) to assign the correct credentials for the version of your database.

Configuring 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 *Oracle Database Backup and Recovery User's Guide*.

3. Enable the local archive destination.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

For more information about these initialization parameters, see *Oracle Data Guard Concepts and Administration*.

Preparing 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](#)
- [Configure the Database to Archive Standby Redo Log Files Locally](#)

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 11g Release 2 (11.2)* for detailed instructions about adding standby redo log files to a database.

 **Note:**

If there will be multiple source databases sending redo to a single downstream mining database, only one of those sources can send redo to the standby redo logs of the mining database. An Extract process that mines the redo from this source database can run in real-time mode. All other source databases must send only their archived logs to the downstream mining database, and the Extracts that read this data must be configured to run in archived-log-only mode.

To Create the Standby Redo Log Files

1. In SQL*Plus, connect to the source database as an administrative user.
2. Determine the size of the source log file. Make note of the results.

```
SELECT BYTES FROM V$LOG;
```

3. Determine the number of online log file groups that are configured on the source database. Make note of the results.

```
SELECT COUNT(GROUP#) FROM V$LOG;
```

4. Connect to the downstream mining database as an administrative user.
5. Add the standby log file groups to the mining database. The standby log file size must be at least the size of the source log file size. The number of standby log file groups must be at least one more than the number of source online log file groups. This applies to each instance (thread) in a RAC installation. So if you have "n" threads at the source database, each having "m" redo log groups, you should configure n*(m+1) redo log groups at the downstream mining database.

The following example shows three standby log groups.

```
ALTER DATABASE ADD STANDBY LOGFILE GROUP 3
('/oracle/dbs/slog3a.rdo', '/oracle/dbs/slog3b.rdo') SIZE 500M;
ALTER DATABASE ADD STANDBY LOGFILE GROUP 4
('/oracle/dbs/slog4.rdo', '/oracle/dbs/slog4b.rdo') SIZE 500M;
ALTER DATABASE ADD STANDBY LOGFILE GROUP 5
('/oracle/dbs/slog5.rdo', '/oracle/dbs/slog5b.rdo') SIZE 500M;
```

6. Confirm that the standby log file groups were added successfully.

```
SELECT GROUP#, THREAD#, SEQUENCE#, ARCHIVED, STATUS
FROM V$STANDBY_LOG;
```

The output should be similar to the following:

GROUP#	THREAD#	SEQUENCE#	ARC	STATUS
3	0	0	YES	UNASSIGNED
4	0	0	YES	UNASSIGNED
5	0	0	YES	UNASSIGNED

7. Ensure that log files from the source database are appearing in the location that is specified in the `LOCATION` attribute of the local `LOG_ARCHIVE_DEST_n` that you set. You might need to switch the log file at the source database to see files in the directory.

Configure the 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
```

C

Example Downstream Mining Configuration

This appendix contains examples for preparing a downstream Oracle mining database to support Extract in integrated capture mode.

Configuring a downstream mining database, see [Configuring a Downstream Mining Database](#) .

Topics:

- [Example 1: Capturing from One Source Database in Real-time Mode](#)
This example captures changes from source database DBMS1 by deploying an integrated capture session at a downstream mining database DBMSCAP.
- [Example 2: Capturing from Multiple Sources in Archive-log-only Mode](#)
The following example captures changes from database DBMS1 and DBMS2 by deploying an integrated capture session at a downstream mining database DBMSCAP.
- [Example 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 integrated capture session at a downstream mining database DBMSCAP.

Example 1: Capturing from One Source Database in Real-time Mode

This example captures changes from source database DBMS1 by deploying an integrated capture session at a downstream mining database DBMSCAP.



Note:

The example assumes that you created the necessary standby redo log files as shown in [Configuring a Downstream Mining Database](#) .

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](#)
- [Prepare the Mining Database to Archive Redo Received in Standby Redo Logs from the Source Database](#)
- [Prepare the Source Database to Send Redo to the Mining Database](#)
- [Set up Integrated Capture \(ext1\) on DBMSCAP](#)

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/sr1_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';
```

```
NAME                VALUE
```

```
-----
compatible      11.1.0.7.0
```

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 Integrated Capture (ext1) on DBMSCAP

To set up integrated capture (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.

```
GGSCI> DBLOGIN USERIDALIAS ggadm1
GGSCI> MININGDBLOGIN USERIDALIAS ggadmcap
GGSCI> REGISTER EXTRACT ext1 DATABASE
```

2. Create Extract at the downstream mining database.

```
GGSCI> 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.

```
GGSCI> START EXTRACT ext1
```

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 capturing changes for database DBMS1 in the preceding example.

Example 2: Capturing from Multiple Sources in Archive-log-only Mode

The following example captures changes from database DBMS1 and DBMS2 by deploying an integrated capture session 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](#)
- [Prepare the Mining Database to Archive Redo from the Source Database](#)
- [Prepare the First Source Database to Send Redo to the Mining Database](#)
- [Prepare the Second Source Database to Send Redo to the Mining Database](#)
- [Set up Extracts at Downstream 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 from the Source Database

Set `DG_CONFIG` at the downstream mining database.

```
ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(dbms1,dbms2, 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';
```

NAME	VALUE
-----	-----
compatible	11.1.0.0.0

The minimum compatibility setting required from integrated 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;
```

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 integrated 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%.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;
```

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.

Example 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 integrated capture session at a downstream mining database DBMSCAP.



Note:

This example assumes that you created the necessary standby redo log files as shown in [Configuring a Downstream Mining Database](#).

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](#)
- [Prepare the Mining Database to Accept Redo from the Source Databases](#)
- [Prepare the First Source Database to Send Redo to the Mining Database](#)
- [Prepare the Second Source Database to Send Redo to the Mining Database](#)
- [Prepare the Third Source Database to Send Redo to the Mining Database](#)
- [Set up Extracts at Downstream 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/
localVALID_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 [Configuring a Downstream Mining Database](#).

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/sr1_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';
```

NAME	VALUE
compatible	11.1.0.0.0

The minimum compatibility setting required from integrated 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;
```

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 integrated 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;
```

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 integrated 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;
```

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.

- [Set up Extract \(ext1\) to Capture Changes from Archived Logs Sent by DBMS1](#)
- [Set up Extract \(ext2\) to Capture Changes from Archived Logs Sent by DBMS2](#)
- [Set up Extract \(ext3\) to Capture Changes in Real-time Mode from Online Logs Sent by DBMS3](#)

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.

```
GGSCI> DBLOGIN USERIDALIAS ggadm1
GGSCI> MININGDBLOGIN USERIDALIAS ggadmcap
GGSCI> REGISTER EXTRACT ext1 DATABASE
```

2. Add Extract at the mining database DBMSCAP.

```
GGSCI> 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.

```
GGSCI> 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`.

```
GGSCI> DBLOGIN USERIDALIAS ggadm2
GGSCI> MININGDBLOGIN USERIDALIAS ggadmcap
GGSCI> REGISTER EXTRACT ext2 DATABASE
```

2. Create Extract at the mining database.

```
GGSCI> 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.

```
GGSCI> 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`.

```
GGSCI> DBLOGIN USERID ggadm3
GGSCI> MININGDBLOGIN USERID ggadmcap
GGSCI> REGISTER EXTRACT ext3 DATABASE
```

2. Create Extract at the mining database.

```
GGSCI> ADD EXTRACT ext3 INTEGRATED TRANLOG, BEGIN NOW
```

3. Edit the Extract parameter file `ext3.prm`. To enable real-time mining, you must specify `downstream_real_time_mine`. 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 Y)
```

4. Start Extract.

```
GGSCI> 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.

D

Installing Trigger-Based DDL Capture

This appendix contains instructions for installing the objects that support the trigger-based method of Oracle GoldenGate DDL support.

To configure Oracle GoldenGate to capture and replicate DDL, see [Configuring DDL Support](#).



Note:

DDL support for sequences (`CREATE`, `ALTER`, `DROP`, `RENAME`) is compatible with, but not required for, replicating the sequence values themselves. To replicate just sequence values, you do not need to install the Oracle GoldenGate DDL support environment. You can just use the `SEQUENCE` parameter in the Extract configuration.

Topics:

- [When to Use Trigger-based DDL Capture](#)
This topic describes the configuration where you must use trigger-based DDL Extract.
- [Overview of the Objects that Support Trigger-based DDL Capture](#)
This topic lists the requirements for installing Oracle GoldenGate trigger-based DDL environment.
- [Installing the DDL Objects](#)
To install DDL objects, you need scripts to perform various tasks during the installation.

When to Use Trigger-based DDL Capture

This topic describes the configuration where you must use trigger-based DDL Extract.

You *must* use trigger-based DDL capture when Extract will operate in the following configurations:

Extract operates in classic capture mode against any version of Oracle Database. If Extract will run in integrated mode against a version 11.2.0.4 or later Oracle Database, the DDL trigger is not required. By default, DDL capture is handled transparently through the database logmining server.

If Extract will capture from a multitenant container database, integrated capture mode must be used with the native DDL capture method.

See [Choosing Capture and Apply Modes](#) for more information about capture modes.

See [Configuring DDL Support](#) for more information about configuring DDL support.

Overview of the Objects that Support Trigger-based DDL Capture

This topic lists the requirements for installing Oracle GoldenGate trigger-based DDL environment.

To install the Oracle GoldenGate trigger-based DDL environment, you will be installing the database objects listed in the following table.

Object	Purpose	Default name
DDL marker table	Stores DDL information. This table only receives inserts.	GGG_MARKER
Sequence on marker table	Used for a column in the marker table.	GGG_DDL_SEQ
DDL history table	Stores object metadata history. This table receives inserts, updates, deletes.	GGG_DDL_HIST
Object ID history table	Contains object IDs of configured objects.	GGG_DDL_HIST_ALT
DDL trigger	Fires on DDL operations. Writes information about the operation to the marker and history tables. Installed with the trigger are some packages.	GGG_DDL_TRIGGER_BEFORE
DDL schema	Contains the DDL synchronization objects.	None; must be specified during installation and in the GLOBALS file.
User role	Establishes the role needed to execute DDL operations.	GGG_GGSUSER_ROLE
Internal setup table	Database table for internal use only.	GGG_SETUP
ddl_pin	Pins DDL tracing, the DDL package, and the DDL trigger for performance improvements.	ddl_pin
ddl_cleartrace.sql	Removes the DDL trace file.	ddl_cleartrace.sql
ddl_status.sql	Verifies that the Oracle GoldenGate DDL objects are installed	ddl_status.sql
marker_status.sql	Verifies that the marker table is installed.	marker_status.sql
ddl_tracelevel.sql	Sets the level for DDL tracing.	ddl_tracelevel.sql

Installing the DDL Objects

To install DDL objects, you need scripts to perform various tasks during the installation.

These scripts are located in the installation directory of Oracle GoldenGate Microservices. The specific location is: `oggma_install_home/lib/sql/legacy`.

Follow these steps to install the database objects that support Oracle GoldenGate DDL capture.



Note:

When using Extract in classic mode to capture in an Active Data Guard environment, the DDL objects must be installed on the source database, not the standby.

1. Choose a schema that can contain the Oracle GoldenGate DDL objects. This schema cannot be case-sensitive.
2. Grant the following permission to the Oracle GoldenGate schema.

```
GRANT EXECUTE ON utl_file TO schema;
```
3. Create a default tablespace for the Oracle GoldenGate DDL schema. This tablespace must be dedicated to the DDL schema; do not allow any other schema to share it.
4. Set `AUTOEXTEND` to `ON` for the DDL tablespace, and size it to accommodate the growth of the `GGG_DDL_HIST` and `GGG_MARKER` tables. The `GGG_DDL_HIST` table, in particular, will grow in proportion to overall DDL activity.
5. (Optional) To cause user DDL activity to fail when the DDL tablespace fills up, edit the `params.sql` script and set the `ddl_fire_error_in_trigger` parameter to `TRUE`. Extract cannot capture DDL if the tablespace fills up, so stopping the DDL gives you time to extend the tablespace size and prevent the loss of DDL capture. Managing tablespace sizing this way, however, requires frequent monitoring of the business applications and Extract to avoid business disruptions. As a best practice, make certain to size the tablespace appropriately in the first place, and set `AUTOEXTEND` to `ON` so that the tablespace does not fill up.



WARNING:

Make a backup of the `params.sql` script before you edit it to preserve its original state.

6. Create a `GLOBALS` file (or edit it, if one exists).

```
EDIT PARAMS ./GLOBALS
```

 **Note:**

EDIT PARAMS creates a simple text file. When you save the file after EDIT PARAMS, it is saved with the name GLOBALS in upper case, without a file extension, at the root of the Oracle GoldenGate directory. Do not alter the file name or location.

7. In the GLOBALS file, specify the name of the DDL schema by adding the following parameter to the GLOBALS file.

```
GGSCHEMA schema_name
```

8. (Optional) To change the names of other objects listed in [DDL synchronization objects](#), *the changes must be made now*, before proceeding with the rest of the installation. Otherwise, you will need to stop Oracle GoldenGate DDL processing and reinstall the DDL objects. It is recommended that you accept the default names of the database objects. To change any database object name (except the schema), do one or both of the following:
 - Record all name changes in the `params.sql` script. Edit this script and change the appropriate parameters. Do not run this script.
 - List the names shown in [Table D-1](#) in the GLOBALS file. The correct parameters to use are listed in the **Parameter** column of the table.

Table D-1 GLOBALS Parameters for Changing DDL Object Names

Object	Parameter
Marker table	MARKERTABLE <i>new_table_name</i> ¹
History table	DDLTABLE <i>new_table_name</i>

¹ Do not qualify the name of any of these tables. The schema name for these table must be either the one that is specified with GGSCHEMA or the schema of the current user, if GGSCHEMA is not specified in GLOBALS.

9. To enable trigger-based DDL replication to recognize Oracle invisible indexes as unique identifiers, set the following parameter to TRUE in the `params.sql` script:

```
define allow_invisible_index_keys = 'TRUE'
```

10. Save and close the GLOBALS file and the `params.sql` file.
11. Change directories to the Oracle GoldenGate installation directory.
12. Exit all Oracle sessions, including those of SQL*Plus, those of business applications, those of the Oracle GoldenGate processes, and those of any other software that uses Oracle. Prevent the start of any new sessions.
13. Run SQL*Plus and log in as a user that has SYSDBA privilege. This privilege is required to install the DDL trigger in the SYS schema, which is required by Oracle. All other DDL objects are installed in the schema that you created.
14. Run the `marker_setup.sql` script. Supply the name of the Oracle GoldenGate schema when prompted, and then press **Enter** to execute the script. The script installs support for the Oracle GoldenGate DDL marker system.

```
@marker_setup.sql
```

15. Run the `ddl_setup.sql` script. You are prompted to specify the name of the DDL schema that you configured. (Note: `ddl_setup.sql` will fail if the tablespace for this schema is shared by any other users. It will not fail, however, if the default tablespace does not have `AUTOEXTEND` set to `ON`, the recommended setting.)

```
@ddl_setup.sql
```

16. Run the `role_setup.sql` script. At the prompt, supply the DDL schema name. The script drops and creates the role that is needed for DDL synchronization, and it grants DML permissions on the Oracle GoldenGate DDL objects.

```
@role_setup.sql
```

17. Grant the role that was created (default name is `GGG_GGSUSER_ROLE`) to all Oracle GoldenGate Extract users. You may need to make multiple grants if the processes have different user names.

```
GRANT role TO user;
```

18. Run the `ddl_enable.sql` script to enable the DDL trigger.

```
@ddl_enable.sql
```

To Install and Use the Optional Performance Tool

To improve the performance of the DDL trigger, make the `ddl_pin` script part of the database startup. It must be invoked with the Oracle GoldenGate DDL user name, as in:

```
@ddl_pin DDL_user
```

This script pins the PL/SQL package that is used by the trigger into memory. If executing this script from SQL*Plus, connect as `SYSDBA` from the Oracle GoldenGate installation directory. This script relies on the Oracle `dmbs_shared_pool` system package, so install that package before using `ddl_pin`.

E

Supporting Changes to XML Schemas

This appendix contains instructions for supporting changes to an XML schema. Both classic and integrated capture modes do 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.
- [Supporting CopyEvolve](#)
The `CopyEvolve` procedure evolves, or changes, a schema and can modify tables by adding or removing columns.

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.

Once 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.

F

Preparing DBFS for an Active-Active Configuration

This appendix contains 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.
- [Applying the Required Patch](#)
Apply the Oracle DBFS patch for bug-9651229 on both databases.
- [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.
- [Partitioning the DBFS Sequence Numbers](#)
DBFS uses an internal sequence-number generator to construct unique names and unique IDs.
- [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.
- [Mapping Local and Remote Peers Correctly](#)
The names of the tables that underlie the DBFS file systems are generated internally and dynamically.

Supported Operations and Prerequisites

This topic lists what is supported by Oracle GoldenGate for DBFS.

Oracle GoldenGate for DBFS supports the following:

- Supported DDL (like `TRUNCATE` or `ALTER`) on DBFS objects except for `CREATE` statements on the DBFS objects. `CREATE` on DBFS must be excluded from the configuration, as must any schemas that will hold the created DBFS objects. The reason to exclude `CREATE` is that the metadata for DBFS must be properly populated in the `SYS` dictionary tables (which itself is excluded from Oracle GoldenGate capture by default).
- Capture and replication of DML on the tables that underlie the DBFS file system.

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

```
declare
begin
dbms_dbfs_sfs_admin.partition_sequence(nodes => 2, myid => 0, newstart
=> :maxval);
commit;
```



```
end;  
/
```

Node 2

```
declare  
begin  
  dbms_dbfs_sfs_admin.partition_sequence( nodes => 2, myid => 1, 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 F-1

```
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 F-2 Node 1

```
declare
dbms_dbfs_content.mountStore('FS1', 'local');
dbms_dbfs_content.mountStore('FS2', 'remote',
read_only => true);
commit;
end;
/
```

Example F-3 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 F-4

```
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 F-5 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 F-6 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 F-7 Node1

```
TABLE [container.]schema.SFS$_FST_100
TABLE [container.]schema.SFS$_FSTP_100;
```

Example F-8 Node2

```
TABLE [container.]schema.SFS$_FST_119
TABLE [container.]schema.SFS$_FSTP_119;
```

Example F-9 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 F-10 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;
```

G

Support for Classic Extract

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

Data type	Classic capture
Scalar columns including DATE and DATETIME columns	Captured from redo.
LONG VARCHAR	Not supported.
BASICFILE LOB columns	LOB modifications done using DML (INSERT/UPDATE/DELETE) are captured from redo. LOB modifications done using DBMS_LOB package are captured from the source table by fetching values from the base table.
SECUREFILE LOB columns	Captured from redo, except for the following cases where SECUREFILE LOBs are fetched from the source table: <ul style="list-style-type: none"> • LOB is encrypted • LOB is compressed • LOB is deduplicated • LOB is stored in-line • LOB is modified using DBMS_LOB package • NOLOGGING LOBs
Index Organized Tables (IOT)	Captured from redo with the following restrictions: <ul style="list-style-type: none"> • IOT with mapping table not supported. • Direct load inserts to IOT tables cannot have the SORTED clause. • IOT with prefix compression as specified with COMPRESS clause is not supported.
XML columns stored as CLOB	Captured from redo.
XML columns stored as Binary	Fetches from source table.
XML columns stored as Object-Relational	Not supported.
XML Type Table	Not supported.
User Defined Type (UDT) columns	Fetches from source table.
Invisible Columns	Not supported.

Data type	Classic capture
ANYDATA columns	<p>Fetches from source table with the following data types only:</p> <p>BINARY_DOUBLE BINARY_FLOAT CHAR DATE INTERVAL DAY TO SECOND INTERVAL YEAR TO MONTH NCHAR NUMBER NVARCHAR2 RAW TIMESTAMP TIMESTAMP WITH TIME ZONE TIMESTAMP WITH LOCAL TIMEZONE</p> <p>UDTs</p> <p>VARCHAR/VARCHAR2</p> <p>Requires source database compatibility to be set to 11.2.0.0.0 or higher.</p>
Spatial Types columns	Fetches from source table.
Collections columns (VARRAYs)	Fetches from source table.
Collections columns (Nested Tables)	<p>Fetches from source table with limitations.</p> <p>See Details of Support for Objects and Operations in Oracle DML.</p>
Object Table	Fetches from source table.
Transparent Data Encryption (Column Encryption & Tablespace Encryption)	Captured from redo.
Basic Compression	Not supported.
OLTP-Compression	Not supported.
Exadata Hybrid Columnar Compression	Not supported.
XA on non-RAC database	Captured from redo.
XA on RAC database	<p>Not supported.</p> <p>To get support, must make sure all branches of XA goes to the same instance.</p>
PDML on non-RAC database	Captured from redo.
PDML on RAC database	<p>Not supported.</p> <p>To get support, you must make sure child transactions spawned from a PDML transaction do not span multiple instances.</p>

- [Details of Support for Objects and Operations in Oracle DML](#)
This section outlines the Oracle objects and operations that Oracle GoldenGate supports for the capture and replication of DML operations.
- [Details of Support for Objects and Operations in Oracle DDL \(Classic\)](#)
This topic outlines the Oracle objects and operation types that Oracle GoldenGate supports for the capture and replication of DDL operations.

Details of Support for Objects and Operations in Oracle DML

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

Supported Objects and Operations in Oracle DML

Identity Columns are supported.

- [Limitations of Support for Index-Organized Tables](#)
- [Limitations of Support for Clustered Tables](#)
- [Non-supported Objects and Operations in Oracle DML \(Classic\)](#)

Limitations of Support for Index-Organized Tables

These limitations apply to classic capture mode.

- IOT with key compression enabled (indicated by the `COMPRESS` keyword in the `key_compression` clause) is not supported in classic capture mode, but is supported in integrated capture mode.

Limitations of Support for Clustered Tables

Indexed clusters are supported by Extract while hash clusters are not supported. In classic capture mode the following limitations apply:

- Encrypted and compressed clustered tables are not supported in classic capture.
- Extract in classic capture mode captures DML changes made to index clustered tables if the cluster size remains the same. Any DDL that causes the cluster size to increase or decrease may cause Extract to capture subsequent DML on that table incorrectly.

Non-supported Objects and Operations in Oracle DML (Classic)

The following are not supported in classic capture:

- Exadata Hybrid Columnar Compression
- Capture from tables with OLTP table compression
- Capture from tablespaces and tables created or altered with `COMPRESS`
- Capture from encrypted and compressed clustered tables
- Invisible column
- Distributed transactions. In Oracle versions 11.1.0.6 and higher, you can capture these transactions if you make them non-distributed by using the following command, which requires the database to be restarted.

```
alter system set _CLUSTERWIDE_GLOBAL_TRANSACTIONS=FALSE;
```

- RAC distributed XA and PDML distributed transactions
- Version enabled-tables

Also see [Non-supported Objects and Operations in Oracle DML](#).

Details of Support for Objects and Operations in Oracle DDL (Classic)

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

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

H

Configuring Capture in Classic Mode

This chapter contains instructions for configuring the Oracle GoldenGate capture process in classic mode.



Note:

To switch an active Extract configuration from integrated to classic mode, perform these configuration steps and then see *Administering Oracle GoldenGate*.

Topics:

- [Prerequisites for Configuring Classic Capture](#)
You must adhere to the guidelines in this topic before configuring an Extract in classic mode.
- [What to Expect from these Instructions](#)
These instructions show you how to configure a basic Extract parameter (configuration) file for the primary Extract, which captures transaction data from the data source, and for a data-pump Extract, which propagates captured data that is stored locally in a *trail* from the source system to the target system.
- [Configuring the Primary Extract in Classic Capture Mode](#)
You can set up a classic Extract process for initial loading of source data and replicating it.
- [Configuring the Data Pump Extract](#)
These steps configure the data pump that reads the local trail and sends the data across the network to a remote trail. The data pump is optional, but recommended.
- [Next Steps](#)
A parameter file is a plain text file that is read by an associated Oracle GoldenGate process to control the product functionality.

Prerequisites for Configuring Classic Capture

You must adhere to the guidelines in this topic before configuring an Extract in classic mode.

The guidelines for configuring Extract in classic mode are:

1. [Preparing the Database for Oracle GoldenGate](#).
2. [Establishing Oracle GoldenGate Credentials](#).
3. [Choosing Capture and Apply Modes](#).
4. Create the Oracle GoldenGate instance on the source system by configuring the Manager process. See *Administering Oracle GoldenGate*.
5. Additionally, review the guidelines in *Administering Oracle GoldenGate*.

What to Expect from these Instructions

These instructions show you how to configure a basic Extract parameter (configuration) file for the primary Extract, which captures transaction data from the data source, and for a data-pump Extract, which propagates captured data that is stored locally in a *trail* from the source system to the target system.

Your business requirements probably will require a more complex topology, but this procedure forms a basis for the rest of your configuration steps.

By performing these steps, you can:

- get the basic configuration files established.
- build upon them later by adding more parameters as you make decisions about features or requirements that apply to your environment.
- use copies of them to make the creation of additional parameter files faster than starting from scratch.



Note:

These instructions do not configure Oracle GoldenGate to perform DDL capture or replication. To support DDL, create the parameter files and then see the following chapters:

[Installing Trigger-Based DDL Capture](#)

[Configuring DDL Support](#)

Configuring the Primary Extract in Classic Capture Mode

You can set up a classic Extract process for initial loading of source data and replicating it.

These steps configure Extract to capture transaction data in classic mode.

1. In GGSCI on the source system, create the Extract parameter file.

```
EDIT PARAMS name
```

Where: *name* is the name of the primary Extract.

2. Enter the Extract parameters in the order shown, starting a new line for each parameter statement.

Basic parameters for the primary Extract in classic capture mode

```
EXTRACT finance  
USERIDALIAS tiger1  
LOGALLSUPCOLS  
ENCRYPTTRAIL AES192  
EXTTRAIL /ggs/dirdat/lt  
SEQUENCE hr.employees_seq;  
TABLE hr.*;
```

Parameter	Description
<code>EXTRACT group</code>	<code>group</code> is the name of the Extract group. For more information, see <i>Reference for Oracle GoldenGate</i> .
<code>USERIDALIAS alias</code>	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, see Establishing Oracle GoldenGate Credentials .
<code>LOGALLSUPCOLS</code>	Writes all supplementally logged columns to the trail, including those required for conflict detection and resolution and the scheduling columns required to support integrated Replicat. (Scheduling columns are primary key, unique index, and foreign key columns.) You configure the database to log these columns with GGSCI commands. See Configuring Logging Properties .
<code>UPDATERECORDFORMAT COMPACT</code>	Combines the before and after images of an <code>UPDATE</code> operation into a single record in the trail. This parameter is valid for Oracle Databases version 12c and later to support Replicat in integrated mode. Although not a required parameter, <code>UPDATERECORDFORMAT COMPACT</code> is a best practice and significantly improves Replicat performance. See <i>Reference for Oracle GoldenGate</i> for more information.
<code>ENCRYPTTRAIL algorithm</code>	Encrypts the local trail.
<code>EXTTRAIL pathname</code>	Specifies the path name of the local trail to which the primary Extract writes captured data. For more information, see <i>Reference for Oracle GoldenGate</i>
<code>{TABLE SEQUENCE} schema.object;</code>	<p>Specifies the database object for which to capture data.</p> <ul style="list-style-type: none"> <code>TABLE</code> specifies a table or a wildcarded set of tables. <code>SEQUENCE</code> specifies a sequence or a wildcarded set of sequences. <code>schema</code> is the schema name or a wildcarded set of schemas. <code>object</code> is the table or sequence name, or a wildcarded set of those objects. <p>See <i>Administering Oracle GoldenGate</i> for information about how to specify object names with and without wildcards.</p> <p>Terminate the parameter statement with a semi-colon.</p> <p>To exclude tables from a wildcard specification, use the <code>TABLEEXCLUDE</code> parameter. See <i>Reference for Oracle GoldenGate</i> for more information about usage and syntax.</p> <p>For more information and for additional <code>TABLE</code> options that control data filtering, mapping, and manipulation, see <i>Reference for Oracle GoldenGate</i>.</p>

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 in GGSCI. For more information, see the *Reference for Oracle GoldenGate*.
4. Save and close the file.

Configuring the Data Pump Extract

These steps configure the data pump that reads the local trail and sends the data across the network to a remote trail. The data pump is optional, but recommended.

The steps to set up the data pump are:

1. In GGSCI on the source system, create the data-pump parameter file.

```
EDIT PARAMS name
```

Where: *name* is the name of the data pump Extract.

2. Enter the data-pump parameters in the order shown, starting a new line for each parameter statement. Your input variables will be different. See [Basic parameters for the data-pump Extract](#) for descriptions.

Basic parameters for the data-pump Extract group using two-part object names:

```
EXTRACT extpump
USERIDALIAS tiger1
RMTHOST fin1, MGRPORT 7809 ENCRYPT AES192, KEYNAME securekey2
RMTRAIL /ggs/dirdat/rt
SEQUENCE hr.employees_seq;
TABLE hr.*;
```

Basic parameters for the data-pump Extract group using three-part object names (including a pluggable database):

```
EXTRACT extpump
USERIDALIAS tiger1
RMTHOST fin1, MGRPORT 7809 ENCRYPT AES192, KEYNAME securekey2
RMTRAIL /ggs/dirdat/rt
TABLE test.ogg.tab1;
SOURCECATALOG pdb1
SEQUENCE hr.employees_seq;
TABLE hr.*;
SOURCECATALOG pdb2
TABLE sales.*;
TABLE acct.*;
```

Parameter	Description
EXTRACT <i>group</i>	<i>group</i> is the name of the data pump Extract. For more information, see <i>Reference for Oracle GoldenGate</i> .
USERIDALIAS <i>alias</i>	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, see Establishing Oracle GoldenGate Credentials .
RMTHOST <i>hostname</i> , MGRPORT <i>portnumber</i> , [, ENCRYPT <i>algorithm</i> KEYNAME <i>keyname</i>]	<ul style="list-style-type: none"> • RMTHOST specifies the name or IP address of the target system. • MGRPORT specifies the port number where Manager is running on the target. • ENCRYPT specifies optional encryption of data across TCP/IP. For additional options and encryption details, see <i>Reference for Oracle GoldenGate</i> .
RMTRAIL <i>pathname</i>	Specifies the path name of the remote trail. For more information, see <i>Reference for Oracle GoldenGate</i> .
SOURCECATALOG <i>container</i>	Use this parameter when the source database is a multitenant container database. Specifies the name of a pluggable database that is to be used as the default container for all subsequent TABLE and SEQUENCE parameters that contain two-part names. This parameter enables you to use two-part object names (<i>schema.object</i>) rather than three-part names (<i>container.schema.object</i>). It remains in effect until another SOURCECATALOG parameter is encountered or a full three-part TABLE or SEQUENCE specification is encountered. Use this parameter when the source database is a multitenant container database. See <i>Reference for Oracle GoldenGate</i> for more information about SOURCECATALOG.

Parameter	Description
<code>{TABLE SEQUENCE}</code> <code>[container.]schema.ob</code> <code>ject;</code>	<p>Specifies a table or sequence, or multiple objects specified with a wildcard. In most cases, this listing will be the same as that in the primary Extract parameter file.</p> <ul style="list-style-type: none"> • <code>TABLE</code> specifies a table or a wildcarded set of tables. • <code>SEQUENCE</code> specifies a sequence or a wildcarded set of sequences. • <code>container</code> is the name of the root container or pluggable database that contains the table or sequence, if this source database is a multitenant container database. See the <code>SOURCECATALOG</code> description in this table. • <code>schema</code> is the schema name or a wildcarded set of schemas. • <code>object</code> is the name of a table or sequence, or a wildcarded set of those objects. <p>See <i>Administering Oracle GoldenGate</i> for information about how to specify object names with and without wildcards.</p> <p>Terminate this parameter statement with a semi-colon.</p> <p>To exclude tables or sequences from a wildcard specification, use the <code>TABLEEXCLUDE</code> or <code>SEQUENCEEXCLUDE</code> parameter after the <code>TABLE</code> statement.</p> <p>For more information and for additional <code>TABLE</code> options that control data filtering, mapping, and manipulation, see <i>Reference for Oracle GoldenGate</i>.</p>

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 in GGSCI. For more information, see the *Reference for Oracle GoldenGate* and [Optional Parameters for Integrated Modes](#) for additional configuration considerations..
4. Save and close the file.

Next Steps

A parameter file is a plain text file that is read by an associated Oracle GoldenGate process to control the product functionality.

Once you have created a basic parameter file for classic capture, see the following for related configuration steps.

[Configuring Oracle GoldenGate Apply](#)

[Additional Oracle GoldenGate Configuration Considerations](#)

[Additional Configuration Steps for Using Classic Capture](#)

[Installing Trigger-Based DDL Capture](#) (to use Oracle GoldenGate DDL support)

[Configuring DDL Support](#) (to use Oracle GoldenGate DDL support)

[Creating Process Groups](#)

[Instantiating Oracle GoldenGate Replication](#)

[Supporting Changes to XML Schemas](#)

Additional Configuration Steps for Using Classic Capture

This chapter contains additional configuration and preparation requirements that are specific only to Extract when operating in *classic capture* mode. These requirements supplement the basic configuration requirements documented in *Configuring Capture in Classic Mode*.

Topics:

- [Configuring Oracle TDE Data in Classic Capture Mode](#)
This section *does not* apply to Extract in integrated capture mode.
- [Using Classic Capture in an Oracle RAC Environment](#)
The following general guidelines apply to Oracle RAC when Extract is operating in classic capture mode.
- [Mining ASM-stored Logs in Classic Capture Mode](#)
This topic covers additional configuration requirements that apply when Oracle GoldenGate mines transaction logs that are stored in Oracle Automatic Storage Management (ASM).
- [Ensuring Data Availability for Classic Capture](#)
To ensure the continuity and integrity of capture processing when Extract operates in classic capture mode, enable archive logging.
- [Configuring Classic Capture in Archived Log Only Mode](#)
You can configure Extract to read exclusively from the archived logs. This is known as **Archived Log Only (ALO)** mode.
- [Configuring Classic Capture in Oracle Active Data Guard Only Mode](#)
You can configure Classic Extract to access both redo data and metadata in real-time to successfully replicate source database activities using Oracle Active Data Guard. This is known as *Active Data Guard* (ADG) mode.
- [Avoiding Log-read Bottlenecks in Classic Capture](#)
When Oracle GoldenGate captures data from the redo logs, I/O bottlenecks can occur because Extract is reading the same files that are being written by the database logging mechanism.

Configuring Oracle TDE Data in Classic Capture Mode

This section *does not* apply to Extract in integrated capture mode.

The following special configuration steps are required to support TDE when Extract is in classic capture mode.

 **Note:**

When in integrated mode, Extract leverages the database logging server and supports TDE column encryption and TDE tablespace encryption without special setup requirements or parameter settings. For more information about integrated capture, see [Choosing Capture and Apply Modes](#).

- [Overview of TDE Support in Classic Capture Mode](#)
- [Requirements for Capturing TDE in Classic Capture Mode](#)
- [Required Database Patches for TDE Support](#)
- [Configuring Classic Capture for TDE Support](#)
- [Recommendations for Maintaining Data Security after Decryption](#)
- [Performing DDL while TDE Capture is Active](#)
- [Rekeying after a Database Upgrade](#)
- [Updating the Oracle Shared Secret in the Parameter File](#)

Overview of TDE Support in Classic Capture Mode

TDE support when Extract is in classic capture mode requires the exchange of two kinds of keys:

- The *encrypted key* can be a table key (column-level encryption), an encrypted redo log key (tablespace-level encryption), or both. This key is shared between the Oracle Database and Extract.
- The *decryption key* is named `ORACLEGG` and its password is known as the *shared secret*. This key is stored securely in the Oracle and Oracle GoldenGate domains. Only a party that has possession of the shared secret can decrypt the table and redo log keys.

The encrypted keys are delivered to the Extract process by means of built-in PL/SQL code. Extract uses the shared secret to decrypt the data. Extract never handles the wallet master key itself, nor is it aware of the master key password. Those remain within the Oracle Database security framework.

Extract never writes the decrypted data to any file other than a trail file, not even a discard file (specified with the `DISCARDFILE` parameter). The word "ENCRYPTED" will be written to any discard file that is in use.

The impact of this feature on Oracle GoldenGate performance should mirror that of the impact of decryption on database performance. Other than a slight increase in Extract startup time, there should be a minimal affect on performance from replicating TDE data.

Requirements for Capturing TDE in Classic Capture Mode

The following are requirements for Extract to support TDE capture:

- To maintain high security standards, the Oracle GoldenGate Extract process should run as part of the `oracle` user (the user that runs the Oracle Database).

That way, the keys are protected in memory by the same privileges as the `oracle` user.

- The Extract process must run on the same machine as the database installation.
- Even if using TDE with a Hardware Security Module, you must use a software wallet. Instructions are provided in [Oracle Security Officer Tasks](#) in the configuration steps for moving from an HSM-only to an HSM-plus-wallet configuration and configuring the `sqlnet.ora` file correctly.
- Whenever the source database is upgraded, you must rekey the master key.

Required Database Patches for TDE Support

To support TDE on Oracle 11.2.0.2, refer to article 1557031.1 on the My Oracle Support website (<https://support.oracle.com>).

Configuring Classic Capture for TDE Support

The following outlines the steps that the Oracle Security Officer and the Oracle GoldenGate Administrator take to establish communication between the Oracle server and the Extract process.

- [Agree on a Shared Secret that Meets Oracle Standards](#)
- [Oracle DBA Tasks](#)
- [Oracle Security Officer Tasks](#)
- [Oracle GoldenGate Administrator Tasks](#)

Agree on a Shared Secret that Meets Oracle Standards

Agree on a *shared secret* password that meets or exceeds Oracle password standards. This password must not be known by anyone else. For guidelines on creating secure passwords, see *Oracle Database Security Guide*.

Oracle DBA Tasks

1. Log in to SQL*Plus as a user with the `SYSDBA` system privilege. For example:

```
sqlplus sys/as sysdba
Connected.
Enter password: password
```

2. Run the `prvtclkm.plb` file that is installed in the Oracle `admin` directory. The `prvtclkm.plb` file creates the `DBMS_INTERNAL_CLKM` PL/SQL package, which enables Oracle GoldenGate to extract encrypted data from an Oracle Database.

```
@?/app/oracle/product/orcl111/rdbms/admin/prvtclkm.plb
```

3. Grant `EXEC` privilege on `DBMS_INTERNAL_CLKM` PL/SQL package to the Extract database user.

```
GRANT EXECUTE ON DBMS_INTERNAL_CLKM TO psmith;
```

4. Exit SQL*Plus.

Oracle Security Officer Tasks

1. Oracle GoldenGate requires the use of a software wallet even with HSM. If you are currently using HSM-only mode, move to HSM-plus-wallet mode by taking the following steps:

- a. Change the `sqlnet.ora` file configuration as shown in the following example, where the wallet directory can be any location on disk that is accessible (rwx) by the owner of the Oracle Database. This example shows a best-practice location, where `my_db` is the `$ORACLE_SID`.

```
ENCRYPTION_WALLET_LOCATION=
(SOURCE=(METHOD=HSM) (METHOD_DATA=
(DIRECTORY=/etc/oracle/wallets/my_db)))
```

- b. Log in to `orapki` (or Wallet Manager) as the owner of the Oracle Database, and create an auto-login wallet in the location that you specified in the `sqlnet.ora` file. When prompted for the wallet password, specify the **same password as the HSM password (or HSM Connect String)**. These two passwords must be identical.

```
cd /etc/oracle/wallets/my_db
orapki wallet create -wallet . -auto_login[_local]
```

 **Note:**

The Oracle Database owner must have full operating system privileges on the wallet.

- c. Add the following entry to the empty wallet to enable an 'auto-open' HSM:

```
mkstore -wrl . -createEntry ORACLE.TDE.HSM.AUTOLOGIN non-empty-string
```

2. Create an entry named `ORACLEGG` in the wallet. **ORACLEGG must be the name of this key.** The password for this key must be the agreed-upon shared secret, but do not enter this password on the command line. Instead, wait to be prompted.

```
mkstore -wrl ./ -createEntry ORACLE.SECURITY.CL.ENCRYPTION.ORACLEGG
Oracle Secret Store Tool : Version 11.2.0.3.0 - Production
Copyright (c) 2004, 2011, Oracle and/or its affiliates. All rights reserved.
Your secret/Password is missing in the command line
Enter your secret/Password: sharedsecret
Re-enter your secret/Password: sharedsecret
Enter wallet password: hsm/wallet_password
```

3. Verify the `ORACLEGG` entry.

```
mkstore -wrl . -list
Oracle Secret Store Tool : Version 11.2.0.3.0 - Production
Copyright (c) 2004, 2011, Oracle and/or its affiliates. All rights reserved.
Enter wallet password: hsm/wallet_password
Oracle Secret Store entries:
ORACLE.SECURITY.CL.ENCRYPTION.ORACLEGG
```

4. Log in to `SQL*Plus` as a user with the `SYSDBA` system privilege.
5. Close and then re-open the wallet.

```
SQL> alter system set encryption wallet close identified by "hsm/wallet_password";
System altered.
SQL> alter system set encryption wallet open identified by "hsm/wallet_password";
System altered.
```

This inserts the password into the auto-open wallet, so that no password is required to access encrypted data with the TDE master encryption key stored in HSM.

6. Switch log files.

```
alter system switch logfile;
System altered.
```

7. If this is an Oracle RAC environment and you are using copies of the wallet on each node, make the copies now and then reopen each wallet.



Note:

Oracle recommends using one wallet in a shared location, with synchronized access among all Oracle RAC nodes.

Oracle GoldenGate Administrator Tasks

1. Run GGSCI.
2. Issue the `ENCRYPT PASSWORD` command to encrypt the shared secret so that it is obfuscated within the Extract parameter file. *This is a security requirement.*

```
ENCRYPT PASSWORD sharedsecret {AES128 | AES192 | AES256} ENCRYPTKEY keyname
```

Where:

- `sharedsecret` is the clear-text shared secret. This value is case-sensitive.
- `{AES128 | AES192 | AES256}` specifies Advanced Encryption Standard (AES) encryption. Specify one of the values, which represents the desired key length.
- `keyname` is the logical name of the encryption key in the `ENCKEYS` lookup file. Oracle GoldenGate uses this key to look up the actual key in the `ENCKEYS` file. To create a key and `ENCKEYS` file, see *Administering Oracle GoldenGate*.

Example:

```
ENCRYPT PASSWORD sharedsecret AES256 ENCRYPTKEY mykey1
```

3. In the Extract parameter file, use the `DBOPTIONS` parameter with the `DECRYPTPASSWORD` option. As input, supply the encrypted shared secret and the decryption key.

```
DBOPTIONS DECRYPTPASSWORD sharedsecret {AES128 | AES192 | AES256} ENCRYPTKEY keyname
```

Where:

- `sharedsecret` is the encrypted shared secret.
- `{AES128 | AES192 | AES256}` must be same value that was used for `ENCRYPT PASSWORD`.
- `keyname` is the logical name of the encryption key in the `ENCKEYS` lookup file.

Example:

```
DBOPTIONS DECRYPTPASSWORD AACAAAAAAAAAAIALCKDZIRHOJBHOJUH AES256
ENCRYPTKEY mykey1
```

4. Log in to SQL*Plus as a user with the SYSDBA system privilege.
5. Close and then re-open the wallet.

```
SQL> alter system set encryption wallet close identified by "hsm/
wallet_password";
System altered.
SQL> alter system set encryption wallet open identified by "hsm/
wallet_password";
System altered.
```

Recommendations for Maintaining Data Security after Decryption

Extract decrypts the TDE data and writes it to the trail as clear text. To maintain data security throughout the path to the target database, it is recommended that you also deploy Oracle GoldenGate security features to:

- encrypt the data in the trails
- encrypt the data in transit across TCP/IP

See ENCRYPTTRAIL | NOENCRYPTTRAIL commands in *Reference for Oracle GoldenGate*.

Performing DDL while TDE Capture is Active

If DDL will ever be performed on a table that has column-level encryption, or if table keys will ever be re-keyed, you must either quiesce the table while the DDL is performed or enable Oracle GoldenGate DDL support. It is more practical to have the DDL environment active so that it is ready, because a re-key usually is a response to a security violation and must be performed immediately. To install the Oracle GoldenGate DDL environment, see [Installing Trigger-Based DDL Capture](#). To configure Oracle GoldenGate DDL support, see [Configuring DDL Support](#). For tablespace-level encryption, the Oracle GoldenGate DDL support is not required.

Rekeying after a Database Upgrade

Whenever the source database is upgraded and Oracle GoldenGate is capturing TDE data, you must rekey the master key, and then restart the database and Extract. The commands to rekey the master key are:

```
alter system set encryption key identified by "mykey";
```

Updating the Oracle Shared Secret in the Parameter File

Use this procedure to update and encrypt the TDE shared secret within the Extract parameter file.

1. Run GGSCI.
2. Stop the Extract process.

```
STOP EXTRACT group
```

3. Modify the `ORACLEGG` entry in the Oracle wallet. `ORACLEGG` must remain the name of the key. For instructions, see *Oracle Database Advanced Security Guide*.
4. Issue the `ENCRYPT PASSWORD` command to encrypt the new shared secret.

```
ENCRYPT PASSWORD sharedsecret {AES128 | AES192 | AES256} ENCRYPTKEY keyname
```

Where:

- `sharedsecret` is the clear-text shared secret. This value is case-sensitive.
- `{AES128 | AES192 | AES256}` specifies Advanced Encryption Standard (AES) encryption. Specify one of the values, which represents the desired key length.
- `keyname` is the logical name of the encryption key in the `ENCKEYS` lookup file.

Example:

```
ENCRYPT PASSWORD sharedsecret AES256 ENCRYPTKEY mykey1
```

5. In the Extract parameter file, use the `DBOPTIONS` parameter with the `DECRYPTPASSWORD` option. As input, supply the encrypted shared secret and the Oracle GoldenGate-generated or user-defined decryption key.

```
DBOPTIONS DECRYPTPASSWORD sharedsecret {AES128 | AES192 | AES256} ENCRYPTKEY  
keyname
```

Where:

- `sharedsecret` is the encrypted shared secret.
- `{AES128 | AES192 | AES256}` must be same value that was used for `ENCRYPT PASSWORD`.
- `keyname` is the logical name of the encryption key in the `ENCKEYS` lookup file.

Example:

```
DBOPTIONS DECRYPTPASSWORD AACAAAAAAAAAAAAIALCKDZIRHOJBHOJUH AES256 ENCRYPTKEY  
mykey1
```

6. Log in to SQL*Plus as a user with the `SYSDBA` system privilege.
7. Close and then re-open the wallet.

```
SQL> alter system set encryption wallet close identified by "hsm/wallet_password";  
System altered.  
SQL> alter system set encryption wallet open identified by "hsm/wallet_password";  
System altered.
```

8. Start Extract.

```
START EXTRACT group
```

Using Classic Capture in an Oracle RAC Environment

The following general guidelines apply to Oracle RAC when Extract is operating in classic capture mode.

- During operations, if the primary database instance against which Oracle GoldenGate is running stops or fails for any reason, Extract abends. To resume processing, you can restart the instance or mount the Oracle GoldenGate binaries to another node where the database is running and then restart the Oracle GoldenGate processes. Stop the

Manager process on the original node before starting Oracle GoldenGate processes from another node.

- Whenever the number of redo threads changes, the Extract group must be dropped and re-created. For the recommended procedure, see *Administering Oracle GoldenGate*.
- Extract ensures that transactions are written to the trail file in commit order, regardless of the RAC instance where the transaction originated. When Extract is capturing in archived-log-only mode, where one or more RAC instances may be idle, you may need to perform archive log switching on the idle nodes to ensure that operations from the active instances are recorded in the trail file in a timely manner. You can instruct the Oracle RDBMS to do this log archiving automatically at a preset interval by setting the `archive_lag_target` parameter. For example, to ensure that logs are archived every fifteen minutes, regardless of activity, you can issue the following command in all instances of the RAC system:


```
SQL> alter system set archive_lag_target 900
```
- To process the last transaction in a RAC cluster before shutting down Extract, insert a dummy record into a source table that Oracle GoldenGate is replicating, and then switch log files on all nodes. This updates the Extract checkpoint and confirms that all available archive logs can be read. It also confirms that all transactions in those archive logs are captured and written to the trail in the correct order.

The following table shows some Oracle GoldenGate parameters that are of specific benefit in Oracle RAC.

Parameter	Description
<code>THREDOPTIONS</code> parameter with the <code>INQUEUESIZE</code> and <code>OUTQUEUESIZE</code> options	Sets the amount of data that Extract queues in memory before sending it to the target system. Tuning these parameters might increase Extract performance on Oracle RAC.
<code>TRANLOGOPTIONS</code> parameter with the <code>PURGEORPHANEDTRANSACTIONS</code> <code>NOPURGEORPHANEDTRANSACTIONS</code> and <code>TRANSCLEANUPFREQUENCY</code> options	Controls how Extract handles orphaned transactions, which can occur when a node fails during a transaction and Extract cannot capture the rollback. Although the database performs the rollback on the failover node, the transaction would otherwise remain in the Extract transaction list indefinitely and prevent further checkpointing for the Extract thread that was processing the transaction. By default, Oracle GoldenGate purges these transactions from its list after they are confirmed as orphaned. This functionality can also be controlled on demand with the <code>SEND EXTRACT</code> command in GGSCI.

Mining ASM-stored Logs in Classic Capture Mode

This topic covers additional configuration requirements that apply when Oracle GoldenGate mines transaction logs that are stored in Oracle Automatic Storage Management (ASM).

- [Accessing the Transaction Logs in ASM](#)
- [Ensuring ASM Connectivity](#)

Accessing the Transaction Logs in ASM

Extract must be configured to read logs that are stored in ASM. Depending on the database version, the following options are available:

- [Reading Transaction Logs Through the RDBMS](#)
- [ASM Direct Connection](#)

Reading Transaction Logs Through the RDBMS

Use the `TRANLOGOPTIONS` parameter with the `DBLOGREADER` option in the Extract parameter file if the RDBMS is Oracle 11.1.0.7 or Oracle 11.2.0.2 or later 11g R2 versions.

An API is available in those releases (but not in Oracle 11g R1 versions) that uses the database server to access the redo and archive logs. When used, this API enables Extract to use a read buffer size of up to 4 MB in size. A larger buffer may improve the performance of Extract when redo rate is high. You can use the `DBLOGREADERBUFSIZE` option of `TRANLOGOPTIONS` to specify a buffer size.



Note:

`DBLOGREADER` also can be used when the redo and archive logs are on regular disk or on a raw device.

When using `DBLOGREADER` and using Oracle Data Vault, grant the `DV_GOLDENGATE_REDO_ACCESS` Role to the Extract database user in addition to the privileges that are listed in [Establishing Oracle GoldenGate Credentials](#).

ASM Direct Connection

If the RDBMS version is not one of those listed in [Reading Transaction Logs Through the RDBMS](#), do the following:

1. Create a user for the Extract process to access the ASM instance directly. Assign this user `SYS` or `SYSDBA` privileges in the ASM instance. Oracle GoldenGate does not support using operating-system authentication for the ASM user.

ASM password configuration ¹	Permitted user
ASM instance and the database share a password file	You can use the Oracle GoldenGate source database user if you grant that user <code>SYSDBA</code> , or you can use any other database user that has <code>SYSDBA</code> privileges.
ASM instance and the source database have separate password files	You can overwrite the ASM password file with the source database password file, understanding that this procedure changes the <code>SYS</code> password in the ASM instance to the value that is contained in the database password file, and it also grants ASM access to the other users in the database password file. Save a copy of the ASM file before overwriting it.

- ¹ To view how the current ASM password file is configured, log on to the ASM instance and issue the following command in SQL*Plus: `SQL> SELECT name, value FROM v$parameter WHERE name = 'remote_login_passwordfile';`
2. Add the ASM user credentials to the Oracle GoldenGate credential store by issuing the `ALTER CREDENTIALSTORE` command. See *Reference for Oracle GoldenGate* for usage instructions and syntax.
3. Specify the ASM login alias in the Extract parameter file by including the `TRANLOGOPTIONS` parameter with the `ASMUSERALIAS` option. For more information about `TRANLOGOPTIONS`, see *Reference for Oracle GoldenGate*.

Ensuring ASM Connectivity

To ensure that the Oracle GoldenGate Extract process can connect to an ASM instance, list the ASM instance in the `tnsnames.ora` file. The recommended method for connecting to an ASM instance when Oracle GoldenGate is running on the database host machine is to use a bequeath (BEQ) protocol. The BEQ protocol does not require a listener. If you prefer to use the TCP/IP protocol, verify that the Oracle listener is listening for new connections to the ASM instance. The `listener.ora` file must contain an entry similar to the following.

```
SID_LIST_LISTENER_ASM =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = ASM)
      (ORACLE_HOME = /u01/app/grid)
      (SID_NAME = +ASM1)
    )
  )
)
```



Note:

A BEQ connection does not work when using a remote Extract configuration. Use `TNSNAMES` with the TCP/IP protocol.

Ensuring Data Availability for Classic Capture

To ensure the continuity and integrity of capture processing when Extract operates in classic capture mode, enable archive logging.

The archive logs provide a secondary data source should the online logs recycle before Extract is finished with them. The archive logs for open transactions must be retained on the system in case Extract needs to recapture data from them to perform a recovery.

⚠ WARNING:

If you cannot enable archive logging, there is a high risk that you will need to completely resynchronize the source and target objects and reinitialize replication should there be a failure that causes an Extract outage while transactions are still active. If you must operate this way, configure the online logs according to the following guidelines to retain enough data for Extract to capture what it needs before the online logs recycle. Allow for Extract backlogs caused by network outages and other external factors, as well as long-running transactions.

In a RAC configuration, Extract must have access to the online and archived logs for all nodes in the cluster, including the one where Oracle GoldenGate is installed.

- [Log Retention Requirements per Extract Recovery Mode](#)
- [Log Retention Options](#)
- [Determining How Much Data to Retain](#)
- [Purging Log Archives](#)
- [Specifying the Archive Location](#)
- [Mounting Logs that are Stored on Other Platforms](#)

Log Retention Requirements per Extract Recovery Mode

The following summarizes the different recovery modes that Extract might use and their log-retention requirements:

- By default, the Bounded Recovery mode is in effect, and Extract requires access to the logs only as far back as twice the Bounded Recovery interval that is set with the `BR` parameter. This interval is an integral multiple of the standard Extract checkpoint interval, as controlled by the `CHECKPOINTSECS` parameter. These two parameters control the Oracle GoldenGate Bounded Recovery feature, which ensures that Extract can recover in-memory captured data after a failure, no matter how old the oldest open transaction was at the time of failure. For more information about Bounded Recovery, see Reference for Oracle GoldenGate.
- In the unlikely event that the Bounded Recovery mechanism fails when Extract attempts a recovery, Extract reverts to normal recovery mode and must have access to the archived log that contains the beginning of the oldest open transaction in memory at the time of failure and all logs thereafter.

Log Retention Options

Depending on the version of Oracle, there are different options for ensuring that the required logs are retained on the system.

- [All Other Oracle Versions](#)

All Other Oracle Versions

For versions of Oracle other than Enterprise Edition, you must manage the log retention process with your preferred administrative tools. Follow the directions in [Determining How Much Data to Retain](#).

Determining How Much Data to Retain

When managing log retention, try to ensure rapid access to the logs that Extract would require to perform a normal recovery (not a Bounded Recovery). See [Log Retention Requirements per Extract Recovery Mode](#). If you must move the archives off the database system, the `TRANLOGOPTIONS` parameter provides a way to specify an alternate location. See [Specifying the Archive Location](#).

The recommended retention period is at least 24 hours worth of transaction data, including both online and archived logs. To determine the oldest log that Extract might need at any given point, issue the `SEND EXTRACT` command with the `SHOWTRANS` option. 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 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 data, and then start the Oracle GoldenGate environment over again.

Purging Log Archives

Make certain not to use backup or archive options that cause old archive files to be overwritten by new backups. Ideally, new backups should be separate files with different names from older ones. This ensures that if Extract looks for a particular log, it will still exist, and it also ensures that the data is available in case it is needed for a support case.

Specifying the Archive Location

If the archived logs reside somewhere other than the Oracle default directory, specify that directory with the `ALTARCHIVELOGDEST` option of the `TRANLOGOPTIONS` parameter in the Extract parameter file.

You might also need to use the `ALTARCHIVEDLOGFORMAT` option of `TRANLOGOPTIONS` if the format that is specified with the Oracle parameter `LOG_ARCHIVE_FORMAT` contains sub-directories. `ALTARCHIVEDLOGFORMAT` specifies an alternate format that removes the sub-directory from the path. For example, `%T/log_%t_%s_%r.arc` would be changed to `log_%t_%s_%r.arc`. As an alternative to using `ALTARCHIVEDLOGFORMAT`, you can create the sub-directory manually, and then move the log files to it.

Mounting Logs that are Stored on Other Platforms

If the online and archived redo logs are stored on a different platform from the one that Extract is built for, do the following:

- NFS-mount the archive files.
- Map the file structure to the structure of the source system by using the `LOGSOURCE` and `PATHMAP` options of the Extract parameter `TRANLOGOPTIONS`.

Configuring Classic Capture in Archived Log Only Mode

You can configure Extract to read exclusively from the archived logs. This is known as **Archived Log Only (ALO)** mode.

In this mode, Extract reads exclusively from archived logs that are stored in a specified location. ALO mode enables Extract to use production logs that are shipped to a secondary database (such as a standby) as the data source. The online logs are not used at all. Oracle GoldenGate connects to the secondary database to get metadata and other required data as needed. As an alternative, ALO mode is supported on the production system.



Note:

ALO mode is not compatible with Extract operating in integrated capture mode.

- [Limitations and Requirements for Using ALO Mode](#)
- [Configuring Extract for ALO mode](#)

Limitations and Requirements for Using ALO Mode

Observe the following limitations and requirements when using Extract in ALO mode.

- Log resets (`RESETLOG`) cannot be done on the source database after the standby database is created.
- ALO cannot be used on a standby database if the production system is Oracle RAC and the standby database is non-RAC. In addition to both systems being Oracle RAC, the number of nodes on each system must be identical.
- ALO on Oracle RAC requires a dedicated connection to the source server. If that connection is lost, Oracle GoldenGate processing will stop.
- It is a best practice to use separate archive log directories when using Oracle GoldenGate for Oracle RAC in ALO mode. This will avoid any possibility of the same file name showing up twice, which could result in Extract returning an "out of order scn" error.
- The `LOGRETENTION` parameter defaults to `DISABLED` when Extract is in ALO mode. You can override this with a specific `LOGRETENTION` setting, if needed.

Configuring Extract for ALO mode

To configure Extract for ALO mode, follow these steps as part of the overall process for configuring Oracle GoldenGate, as documented in [Configuring Capture in Classic Mode](#).

1. Enable supplemental logging at the table level and the database level for the tables in the source database. (See [Configuring Logging Properties](#).)
2. When Oracle GoldenGate is running on a different server from the source database, make certain that SQL*Net is configured properly to connect to a remote server, such as providing the correct entries in a `TNSNAMES` file. Extract must have permission to maintain a SQL*Net connection to the source database.
3. Use a SQL*Net connect string for the name of the user in the credential store that is assigned to the process. Specify the alias of this user in the following:
 - The `USERIDALIAS` parameter in the parameter file of every Oracle GoldenGate process that connects to that database.
 - The `USERIDALIAS` portion of the `DBLOGIN` command in `GGSCI`.

 **Note:**

If you have a standby server that is local to the server that Oracle GoldenGate is running on, you do not need to use a connect string for the user specified in `USERIDALIAS`. You can just supply the user login name.

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

4. Use the Extract parameter `TRANLOGOPTIONS` with the `ARCHIVEDLOGONLY` option. This option forces Extract to operate in ALO mode against a primary or logical standby database, as determined by a value of `PRIMARY` or `LOGICAL STANDBY` in the `db_role` column of the `v$database` view. The default is to read the online logs. `TRANLOGOPTIONS` with `ARCHIVEDLOGONLY` is not needed if using ALO mode against a physical standby database, as determined by a value of `PHYSICAL STANDBY` in the `db_role` column of `v$database`. Extract automatically operates in ALO mode if it detects that the database is a physical standby.
5. Other `TRANLOGOPTIONS` options might be required for your environment. For example, depending on the copy program that you use, you might need to use the `COMPLETEARCHIVEDLOGONLY` option to prevent Extract errors.
6. Use the `MAP` parameter for Extract to map the table names to the source object IDs..
7. Add the Extract group by issuing the `ADD EXTRACT` command with a timestamp as the `BEGIN` option, or by using `ADD EXTRACT` with the `SEQNO` and `RBA` options. It is best to give Extract a known start point at which to begin extracting data, rather than by using the `NOW` argument. The start time of `NOW` corresponds to the time of the current *online* redo log, but an ALO Extract cannot read the online logs, so it must wait for that log to be archived when Oracle switches logs. The timing of the

switch depends on the size of the redo logs and the volume of database activity, so there might be a lag between when you start Extract and when data starts being captured. This can happen in both regular and RAC database configurations.

Configuring Classic Capture in Oracle Active Data Guard Only Mode

You can configure Classic Extract to access both redo data and metadata in real-time to successfully replicate source database activities using Oracle Active Data Guard. This is known as *Active Data Guard (ADG) mode*.

ADG mode enables Extract to use production logs that are shipped to a standby database as the data source. The online logs are not used at all. Oracle GoldenGate connects to the standby database to get metadata and other required data as needed.

This mode is useful in load sensitive environments where ADG is already in place or can be implemented. It can also be used as cost effective method to implement high availability using the ADG Broker role planned (switchover) and failover (unplanned) changes. In an ADG configuration, switchover and failover are considered *roles*. When either of the operations occur, it is considered a *role change*. For more information, see Oracle Data Guard Concepts and Administration and Oracle Data Guard Broker.

You can configure Integrated Extract to fetch table data and metadata required for the fetch from an ADG instead of the source database. This is possible because an ADG is a physical replica of the source database. Fetching from an ADG using the `FETCHUSER` parameter is supported by Extract in all configurations except when running as Classic Extract. Classic Extract already has the ability to connect directly to an ADG and mine its redo logs and fetch from it using standard connection information supplied using the `USERID` parameter. The impact to the source database is minimized because Extract gathers information from the source database at startup, including compatibility level, database type, and source database validation checks, when fetching from an ADG.

All previous fetch functionality and parameters are supported.



Note:

Integrated Extract cannot capture from a standby database because it requires `READ` and `WRITE` access to the database, and an ADG standby only provides `READ ONLY` access.

- [Limitations and Requirements for Using ADG Mode](#)
- [Configuring Classic Extract for ADG Mode](#)
- [Migrating Classic Extract To and From an ADG Database](#)
- [Handling Role Changes In an ADG Configuration](#)

Limitations and Requirements for Using ADG Mode

Observe the following limitations and requirements when using Extract in ADG mode.

- Extract in ADG mode will only apply redo data that has been applied to the standby database by the apply process. If Extract runs ahead of the standby database, it will wait for the standby database to catch up.
- You must explicitly specify ADG mode in your classic Extract parameter file to run extract on the standby database.
- You must specify the database user and password to connect to the ADG system because fetch and other metadata resolution occurs in the database.
- The number of redo threads in the standby logs in the standby database must match the number of nodes from the primary database.
- No new RAC instance can be added to the primary database after classic Extract has been created on the standby database. If you do add new instances, the redo data from the new thread will not be captured by classic Extract.
- Archived logs and standby redo logs accessed from the standby database will be an exact duplicate of the primary database. The size and the contents will match, including redo data, transactional data, and supplemental data. This is guaranteed by a properly configured ADG deployment.
- ADG role changes are infrequent and require user intervention in both cases.
- With a switchover, there will be an indicator in the redo log file header (end of the redo log or EOR marker) to indicate end of log stream so that classic Extract on the standby can complete the RAC coordination successfully and ship all of the committed transactions to the trail file.
- With a failover, a new incarnation is created on both the primary and the standby databases with a new incarnation ID, `RESETLOG` sequence number, and SCN value.
- You must connect to the primary database from GGSCI to add `TRANDATA` or `SCHEMATRANDATA` because this is done on the primary database.
- DDL triggers cannot be used on the standby database, in order to support DDL replication (except `ADDTRANDATA`). You must install the Oracle GoldenGate DDL package on the primary database.
- DDL `ADDTRANDATA` is not supported in ADG mode; you must use `ADDSCHEMATRANDATA` for DDL replication.
- When adding extract on the standby database, you must specify the starting position using a specific SCN value, timestamp, or log position. Relative timestamp values, such as `NOW`, become ambiguous and may lead to data inconsistency.
- When adding extract on the standby database, you must specify the number of threads that will include all of the relevant threads from the primary database.
- During or after failover or switchover, no thread can be added or dropped from either primary or standby databases.
- Classic Extract will only use one intervening `RESETLOG` operation.
- If you do not want to relocate your Oracle GoldenGate installation, then you must position it in a shared space where the Oracle GoldenGate installation directory can be accessed from both the primary and standby databases.
- If you are moving capture off of an ADG standby database to a primary database, then you must point your net alias to the primary database and you must remove the `TRANLOG` options.

- Only Oracle Database releases that are running with compatibility setting of 10.2 or higher (10g Release 2) are supported.
- Classic Extract does not support the `DBLOGREADER` option. Use `ASMUSER` (there is approximately a 20gb/hr read limit) or move the online and archive logs outside of the Application Security Manager (ASM) on both the primary and the standby databases.

 **Note:**

The combination of `MINEFROMACTIVEDG` and `DBLOGREADER` options is not supported with Classic Extract. However, the Extract process will start without any warning or error even though this combination is used. Ensure that you do not use this combination while using classic Extract with ADG.

Configuring Classic Extract for ADG Mode

To configure Classic Extract for ADG mode, follow these steps as part of the overall process for configuring Oracle GoldenGate, as documented in [Configuring Capture in Classic Mode](#).

1. Enable supplemental logging at the table level and the database level for the tables in the *primary* database using the `ADD SCHEMATRANDATA` parameter. If necessary, create a DDL capture.)
2. When Oracle GoldenGate is running on a different server from the source database, make certain that SQL*Net is configured properly to connect to a remote server, such as providing the correct entries in a `TNSNAMES` file. Extract must have permission to maintain a SQL*Net connection to the source database.
3. On the *standby* database, use the Extract parameter `TRANLOGOPTIONS` with the `MINEFROMACTIVEDG` option. This option forces Extract to operate in ADG mode against a standby database, as determined by a value of `PRIMARY` or `LOGICAL STANDBY` in the `db_role` column of the `v$database` view.

Other `TRANLOGOPTIONS` options might be required for your environment. For example, depending on the copy program that you use, you might need to use the `COMPLETEARCHIVEDLOGONLY` option to prevent Extract errors.

4. On the *standby* database, add the Extract group by issuing the `ADD EXTRACT` command specifying the number of threads active on the *primary* database at the given SCN. The timing of the switch depends on the size of the redo logs and the volume of database activity, so there might be a limited lag between when you start Extract and when data starts being captured. This can happen in both regular and RAC database configurations.

Migrating Classic Extract To and From an ADG Database

You must have your parameter files, checkpoint files, bounded recovery files, and trail files stored in shared storage or copied to the ADG database before attempting to migrate a classic Extract to or from an ADG database. Additionally, you must ensure that there has not been any intervening role change or Extract will mine the same branch of redo.

Use the following steps to move to an ADG database:

1. Edit the parameter file `ext1.prm` to add the following parameters:

```
DBLOGIN USERID userid@ADG PASSWORD password
TRANLOGOPTIONS MINEFROMACTIVEDG
```

2. Start Extract by issuing the `START EXTRACT ext1` command.

Use the following steps to move from an ADG database:

1. Edit the parameter file `ext1.prm` to remove the following parameters:

```
DBLOGIN USERID userid@ADG PASSWORD password
TRANLOGOPTIONS MINEFROMACTIVEDG
```

2. Start Extract by issuing the `START EXTRACT ext1` command.

Handling Role Changes In an ADG Configuration

In a role change involving a standby database, all sessions in the primary and the standby database are first disconnected including the connections used by Extract. Then both databases are shut down, then the original primary is mounted as a standby database, and the original standby is opened as the primary database.

The procedure for a role change is determined by the initial deployment of Classic Extract and the deployment relation that you want, database or role. The following table outlines the four possible role changes and is predicated on an ADG configuration comprised of two databases, `prisys` and `stansys`. The `prisys` system contains the primary database and the `stansys` system contains the standby database; `prisys` has two redo threads active, whereas `stansys` has four redo threads active.

Initial Deployment Primary (<code>prisys</code>)	Initial Deployment ADG (<code>stansys</code>)
Original Deployment:	
<code>ext1.prm</code> DBLOGIN USERID <i>userid</i> @prisys, PASSWORD <i>password</i>	<code>ext1.prm</code> DBLOGIN USERID <i>userid</i> @stansys, PASSWORD <i>password</i> TRANLOGOPTIONS MINEFROMACTIVEDG
Database Related:	

Initial Deployment Primary (prisys)	Initial Deployment ADG (stansys)
<u>After Role Transition: Classic Extract to ADG</u>	<u>After Role Transition: ADG to classic Extract</u>
<ol style="list-style-type: none"> <li data-bbox="365 315 852 546">1. Edit the <code>ext1.prm</code> file to add: <code>TRANLOGOPTIONS MINEFROMACTIVEDG</code> <code>DBLOGREADER</code> option cannot be used in ADG mode. If <code>DBLOGREADER</code> option exists, remove it. If using ASM, add the <code>ASMUSER</code> parameter to connect to the ASM instance. <li data-bbox="365 556 852 619">2. If a failover, add <code>TRANLOGOPTIONS USEPREVRESETLOGSID</code>. <li data-bbox="365 630 852 861">3. Start Extract: <code>START EXTRACT ext1</code> Extract will abend once it reaches the role transition point, then it does an internal <code>BR_RESET</code> and moves both the I/O checkpoint and current checkpoint to <code>SCN s</code>. <li data-bbox="365 871 852 1008">4. If failover, edit the parameter file again and remove: <code>TRANLOGOPTIONS</code> <code>USEPREVRESETLOGSID</code> <li data-bbox="365 1018 852 1113">5. Execute <code>ALTER EXTRACT ext1 SCN #</code>, where <code>#</code> is the SCN value from role switch message. <li data-bbox="365 1123 852 1560">6. Based on the thread counts, do one of the following: If the thread counts are same between the databases, then execute the <code>START EXTRACT ext1;</code> command. or If thread counts are different between the databases, then execute the following commands: <code>DROP EXTRACT ext1</code> <code>ADD EXTRACT ext1 THREADS t BEGIN SCN s</code> <code>START EXTRACT ext1</code> 	<ol style="list-style-type: none"> <li data-bbox="852 315 1380 388">1. Edit <code>ext1.prm</code> and remove: <code>TRANLOGOPTIONS MINEFROMACTIVEDG</code> <li data-bbox="852 399 1380 472">2. If a failover, add <code>TRANLOGOPTIONS USEPREVRESETLOGSID</code>. <li data-bbox="852 483 1380 714">3. Start Extract: <code>START EXTRACT ext1</code> Extract will abend once it reaches the role transition point, then it does an internal <code>BR_RESET</code> and moves both the I/O checkpoint and current checkpoint to <code>SCN s</code>. <li data-bbox="852 724 1380 819">4. If failover, edit the parameter file again and remove: <code>TRANLOGOPTIONS USEPREVRESETLOGSID</code> <li data-bbox="852 829 1380 924">5. Execute <code>ALTER EXTRACT ext1 SCN #</code>, where <code>#</code> is the SCN value from role switch message. <li data-bbox="852 934 1380 1560">6. Based on the thread counts, do one of the following: If the thread counts are same between the databases, then execute the <code>START EXTRACT ext1;</code> command. or If thread counts are different between the databases, then execute the following commands: <code>DROP EXTRACT ext1</code> <code>ADD EXTRACT ext1 THREADS t BEGIN SCN s</code> <code>START EXTRACT ext1</code>
Role Related:	

Initial Deployment Primary (prisys)	Initial Deployment ADG (stansys)
<p><u>After Role Transition: Classic Extract to classic Extract</u></p> <ol style="list-style-type: none"> 1. Edit <code>ext1.prm</code> to change the database system to the standby system: <pre>DBLOGIN USERID userid@stansys, PASSWORD password</pre> 2. If a failover, add <code>TRANLOGOPTIONS USEPREVRESETLOGSID</code>. 3. Start Extract: <pre>START EXTRACT ext1</pre> <p>Extract will abend once it reaches the role transition point, then it does an internal <code>BR_RESET</code> and moves both the I/O checkpoint and current checkpoint to SCN s.</p> 4. If failover, edit the parameter file again and remove: <pre>TRANLOGOPTIONS USEPREVRESETLOGSID</pre> 5. Execute <code>ALTER EXTRACT ext1 SCN #</code>, where # is the SCN value from role switch message. 6. Based on the thread counts, do one of the following: <p>If the thread counts are same between the databases, then execute the <code>START EXTRACT ext1; command</code>.</p> <p>or</p> <p>If thread counts are different between the databases, then execute the following commands: <pre>DROP EXTRACT ext1 ADD EXTRACT ext1 THREADS t BEGIN SCN s START EXTRACT ext1</pre> </p> 	<p><u>After Role Transition: ADG to ADG</u></p> <ol style="list-style-type: none"> 1. Edit <code>ext1.prm</code> to change the database system to the primary system: <pre>DBLOGIN USERID userid@prisys, PASSWORD password</pre> 2. If a failover, add <code>TRANLOGOPTIONS USEPREVRESETLOGSID</code>. 3. Start Extract: <pre>START EXTRACT ext1</pre> <p>Extract will abend once it reaches the role transition point, then it does an internal <code>BR_RESET</code> and moves both the I/O checkpoint and current checkpoint to SCN s.</p> 4. If failover, edit the parameter file again and remove: <pre>TRANLOGOPTIONS USEPREVRESETLOGSID</pre> 5. Execute <code>ALTER EXTRACT ext1 SCN #</code>, where # is the SCN value from role switch message. 6. Based on the thread counts, do one of the following: <p>If the thread counts are same between the databases, then execute the <code>START EXTRACT ext1; command</code>.</p> <p>or</p> <p>If thread counts are different between the databases, then execute the following commands: <pre>DROP EXTRACT ext1 ADD EXTRACT ext1 THREADS t BEGIN SCN s START EXTRACT ext1</pre> </p>

Avoiding Log-read Bottlenecks in Classic Capture

When Oracle GoldenGate captures data from the redo logs, I/O bottlenecks can occur because Extract is reading the same files that are being written by the database logging mechanism.

Performance degradation increases with the number of Extract processes that read the same logs. You can:

- Try using faster drives and a faster controller. Both Extract and the database logging mechanism will be faster on a faster I/O system.

- Store the logs on RAID 0+1. Avoid RAID 5, which performs checksums on every block written and is not a good choice for high levels of continuous I/O.