Oracle® Database

Using Oracle GoldenGate Classic Architecture for Non-Oracle Databases

21c
F41982-11
March 2024

Oracle Database Using Oracle GoldenGate Classic Architecture for Non-Oracle Databases, 21c

F41982-11

Primary Author: Oracle Corp.

Contents

6 Configuring Oracle GoldenGate for DB2 for i

7 Instantiating and Starting Oracle GoldenGate Replication

Preface

This guide helps you get started with using Oracle GoldenGate on non-Oracle database systems supported with this release.

Topics:

Audience

Using Oracle GoldenGate for Heterogeneous Databases is intended for database and system administrators who are responsible for implementing Oracle GoldenGate and managing the databases for an organization.

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
- **Oracle GoldenGate Classic for Big Data** in Using Oracle GoldenGate on Oracle Cloud Marketplace

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, such as "From the File menu, select **Save**." Boldface also is used for terms defined in text or in the glossary.
italic `*italic*`	Italic type indicates placeholder variables for which you supply particular values, such as in the parameter statement: `TABLE *table_name*`. Italic type also is used for book titles and emphasis.
`monospace` `MONOSPACE`	Monospace type indicates code components such as user exits and scripts; the names of files and database objects; URL paths; and input and output text that appears on the screen. Uppercase monospace type is generally used to represent the names of Oracle GoldenGate parameters, commands, and user-configurable functions, as well as SQL commands and keywords.
UPPERCASE	Uppercase in the regular text font indicates the name of a process or utility unless the name is intended to be a specific case. Keywords in upper case (`ADD EXTRACT`, `ADD EXTTRAIL`, `FORMAT RELEASE`).
LOWERCASE	Names of processes to be written in lower case. Examples: `ADD EXTRACT exte`, `ADD EXTRAIL ea`.
{ }	Braces within syntax enclose a set of options that are separated by pipe symbols, one of which must be selected, for example: `{*option1* | *option2* | *option3*}`.
[ ]	Brackets within syntax indicate an optional element. For example in this syntax, the `SAVE` clause is optional: `CLEANUP REPLICAT *group_name* [, SAVE *count*]`. Multiple options within an optional element are separated by a pipe symbol, for example: `[*option1* | *option2*]`.
Sample Locations	Compass directions such as east, west, north, south to be used for demonstrating Extract and Replicat locations. Datacenters names to use the standard similar to `dc1`, `dc2`.
Group names	Prefixes for each process, as follows: • Extract: ext. Usage with location: `extn`, where ***n*** indicates 'north' compass direction. • Replicat: rep. Usage with location: `repn`, where ***n*** indicates 'north' compass direction. • Distribution Path: `dp`. Usage with location: `dpn`, where ***n*** indicates 'north' compass direction. • Checkpoint table: `ggs_checkpointtable` • Trail file names: e or d depending on whether the trail file is for the Extract of distribution path. Suffix derived in alphabetical order. Usage for an Extract trail file: `ea`, `eb`, `ec`. • Trail file subdirectory: The name will use compass directions to refer to the trail subdirectories. Example for trail subdirectory name would be `/east`, `/west`, `/north`, `/south`.

Part I
Using Oracle GoldenGate for DB2 LUW

Oracle GoldenGate for DB2 LUW supports capture and delivery of initial load and transactional data for supported DB2 LUW database versions.

Oracle GoldenGate for DB2 LUW supports the mapping, filtering, and transformation of source data, unless noted otherwise in this document, as well as replicating data derived from other source databases supported by Oracle GoldenGate, into DB2 LUW databases.

This part describes tasks for configuring and running Oracle GoldenGate for DB2 LUW.

- Understanding What's Supported for DB2 LUW
  This chapter contains information on database and table features supported by Oracle GoldenGate for DB2 LUW.
- Preparing the System for Oracle GoldenGate
- Configuring Oracle GoldenGate for DB2 LUW

1

Understanding What's Supported for DB2 LUW

This chapter contains information on database and table features supported by Oracle GoldenGate for DB2 LUW.

Topics:

Supported DB2 LUW Data Types

Oracle GoldenGate supports all DB2 LUW data types, except those listed in Non-Supported DB2 LUW Data Types.

Limitations of Support

Oracle GoldenGate has the following limitations for supporting DB2 LUW data types:

- Oracle GoldenGate supports multi-byte character data types and multi-byte data stored in character columns. Multi-byte data is only supported in a like-to-like configuration. Transformation, filtering, and other types of manipulation are not supported for multi-byte character data.
- `BLOB` and `CLOB` columns must have a `LOGGED` clause in their definitions.
- Due to limitations in the IBM DB2READLOG interface, Oracle GoldenGate does not support coordination of transactions across nodes in a DB2 Database Partitioning Feature (DPF) environment. In DPF, a transaction may span multiple nodes, depending upon how the data is partitioned.

  However, if you need to capture from it, you can do it with certain limitations. Check the Oracle Support note Does Oracle GoldenGate Support DB2 LUW Data Partitioning Feature (DPF)? (DocID 2763006.1)

  .
- `GRAPHIC` and `VARGRAPHIC` columns must be in a database, where the character set is UTF16. Any other character set causes the Oracle GoldenGate to abend.
- The support of range and precision for floating-point numbers depends on the host machine. In general, the precision is accurate to 16 significant digits, but you should review the database documentation to determine the expected approximations. Oracle GoldenGate rounds or truncates values that exceed the supported precision.

- Extract fully supports the capture and apply of `TIMESTAMP(0)` through `TIMESTAMP(12)` when the output trail format is 19.1 or higher. Otherwise Extract truncates data from `TIMESTAMP(10)` through `TIMESTAMP(12)` to nanoseconds (maximum of nine digits of fractional time) and issues a warning to the report file.
- Oracle GoldenGate supports timestamp data from 0001/01/03:00:00:00 to 9999/12/31:23:59:59. If a timestamp is converted from GMT to local time, these limits also apply to the resulting timestamp. Depending on the timezone, conversion may add or subtract hours, which can cause the timestamp to exceed the lower or upper supported limit.
- Oracle GoldenGate does not support the filtering, column mapping, or manipulation of large objects that are larger than 4K. You can use the full Oracle GoldenGate functionality for objects that are 4K or smaller.
- Replication of XML columns between source and target databases with the *same* character set is supported. If the source and target database character sets are different, then XML replication may fail with a database error because some characters may not be recognized (or valid) in the target database character set.

Non-Supported DB2 LUW Data Types

The non-supported DB2 LUW data types are:

- `XMLType`
- User-defined types
- Negative dates

Supported Objects and Operations for DB2 LUW

Object and operations that are supported for DB2 LUW are:

- Oracle GoldenGate Extract supports cross-endian capture where the database and Oracle GoldenGate are running on different byte order servers. The byte order is detected automatically for DB2 LUW version 10.5 or higher. If the DB2 database auto-detection on the DB2 LUW 10.5 database is not required then you can override it by specifying the `TRANLOGOPTIONS MIXEDENDIAN [ON|OFF]` parameter. For DB2 LUW version 10.1, this parameter must be used in the Extract parameter file for cross-endian capture. See `TRANLOGOPTIONS` in *Reference for Oracle GoldenGate*.
- DB2 pureScale environment is supported.
- Oracle GoldenGate supports the maximum number of columns and column size per table that is supported by the database.
- `TRUNCATE TABLE`.
- Multi-Dimensional Clustered Tables (MDC).
- Materialized Query Tables. Oracle GoldenGate does not replicate the MQT itself, but only the base tables. The target database automatically maintains the content of the MQT based on the changes that are applied to the base tables by Replicat.
- Tables with `ROW COMPRESSION`. In DB2 LUW version 10.1 and later, `COMPRESS YES STATIC` is supported and `COMPRESS YES ADAPTIVE` are supported.

- Extended row size feature is enabled by default. It is supported with a workaround using `FETCHCOLS`. For any column values that are `VARCHAR` or `VARGRAPHIC` data types and are stored out of row in the database, you must fetch these extended rows by specifying these columns using the `FETCHCOLS` option in the `TABLE` parameter in the extract parameter file. With this option set, when the column values are out of row then Oracle GoldenGate will fetch its value. If the value is out of and `FETCHCOLS` is *not* specified then Extract will abend to prevent any data loss. If you do not want to use this feature, set the `extended_row_size` parameter to `DISABLE`.

  Extended row size feature is enabled, by default. It is supported with a workaround using `FETCHCOLS` for DB2 LUW 10.1. For any column values that are `VARCHAR` or `VARGRAPHIC` data types and are stored out of row in the database, you must fetch these extended rows by specifying these columns using the `FETCHCOLS` option in the `TABLE` parameter in the Extract parameter file. With this option set, when the column values are out of row, then Oracle GoldenGate fetches its value. If the value is out of and `FETCHCOLS` is not specified then Extract abends to prevent any data loss. If you do not want to use this feature, set the `extended_row_size parameter` to `DISABLE`. For DB2 LUW 10.5 and higher out of row values are captured seamlessly by Extract. `FETCHCOLS` is no more needed to capture out of row columns from these database versions.
- Temporal tables with DB2 LUW 10.1 FixPack 2 and greater are supported. This is the default for Replicat.
- Supported options with `SHOWTRANS`

  ```
  SHOWTRANS [transaction_ID]  [COUNT n]
  [DURATION duration unit]
  [TABULAR][FILE file_name] |
  ```

- Options with `SKIPTRANS` and `FORCETRANS`.

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

- Limitations on Automatic Heartbeat Table support are as follows:
  - `[THREAD n] [DETAIL]` is not supported.
  - Oracle GoldenGate heartbeat parameters frequency and purge frequency are accepted in seconds and days. However, the DB2 LUW task scheduler accepts its schedule only in cron format so the Oracle GoldenGate input value to cron format may result in some loss of accuracy. For example:

    ```
    ADD HEARTBEATTABLE, FREQUENCY 150, PURGE_FREQUENCY 20
    ```

    This example sets the `FREQUENCY` to 150 seconds, which is converted to the closest minute value of 2 minutes, so the heartbeat table is updated every 120 seconds instead of every 150 seconds. Setting `PURGE_FREQUENCY` to 20 means that the history table is purged at midnight on every 20th day.
  - The following are steps are necessary for the heartbeat scheduled tasks to run:
    1. Set the `DB2_ATS_ENABLE` registry variable to `db2set DB2_ATS_ENABLE=YES`.
    2. Create the `SYSTOOLSPACE` tablespace if it does not already exist:

       ```
       CREATE TABLESPACE SYSTOOLSPACE IN IBMCATGROUP MANAGED BY AUTOMATIC STORAGE
       EXTENTSIZE 4
       ```

3. Ensure instance owner has Database administration authority (DBADM):

```
GRANT DBADM ON DATABASE TO instance_owner_name
```

Non-Supported Objects and Operations for DB2 LUW

Objects and operations for DB2 LUW that are not supported by Oracle GoldenGate are:

- Schema, table or column names that have trailing spaces
- Multiple instances of a database
- Datalinks
- Extraction or replication of DDL (data definition language) operations
- Generated columns (`GENERATE ALWAYS` clause)

> **Note:**

- DB2 Data Partitioning Feature (DPF) is not supported. DPF doesn't provide a way to read the log records in a coordinated fashion across all the nodes in a partition. So, there is no way to ensure that even if all nodes are being replicated with separate Extracts, it would be possible to ensure that all records from all transactions are applied in the correct temporal order. This may occur due to a number of factors including caching in the database and the underlying operating system not allowing the Extracts to have visibility into whether there are any cached and not yet visible entries that may affect the ordering of the record operations across all partitions. Due to this uncertainty, it is not possible to ensure that transactions that span partitions, or primary key updates can be replicated in a consistent manner.

System Schemas

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

- `"SYSIBM"`
- `"SYSCAT"`
- `"SYSSTAT"`
- `"SYSPROC"`
- `"SYSFUN"`
- `"SYSIBMADMIN"`
- `"SYSTOOLS"`
- `"SYSPUBLIC"`

Supported Object Names

For a list of characters that are supported in object names, see Supported Database Object Names in *Administering Oracle GoldenGate*.

2
Preparing the System for Oracle GoldenGate

This chapter describes how to prepare the environment to run Oracle GoldenGate on DB2 LUW.

Topics:

Configuring the Transaction Logs for Oracle GoldenGate

To capture DML operations, Oracle GoldenGatereads the DB2 LUW online logs by default. However, it reads the archived logs if an online log is not available. To ensure the continuity and integrity of Oracle GoldenGateprocessing, configure the logs as follows.

Retaining the Transaction Logs

Configure the database to retain the transaction logs for roll forward recovery by enabling one of the following parameter sets, depending on the database version.

- DB2 LUW 9.5 and later:

  Set the `LOGARCHMETH` parameters as follows:

  – Set `LOGARCHMETH1` to `LOGRETAIN`.

  – Set `LOGARCHMETH2` to `OFF`.

  Alternatively, you can use any other `LOGARCHMETH` options, as long as forward recovery is enabled. For example, the following is valid:

  – Set `LOGARCHMETH1` to `DISK`.

  – Set `LOGARCHMETH2` to `TSM`.

To determine the log retention parameters:

1. Connect to the database.

   ```
   db2 connect to database user username using password
   ```

2. Get the database name.

```
db2 list db directory
```

3. Get the database configuration for the database.

   ```
   db2 get db cfg for database
   ```

   The fields to view are:

   ```
   Log retain for recovery status = RECOVERY
   User exit for logging status = YES
   ```

To set the log retention parameters:

1. Issue one of the following commands.

   To enable `USEREXIT`:

   ```
   db2 update db cfg for database using USEREXIT ON
   ```

   If not using `USEREXIT`, use this command:

   ```
   db2 update db cfg for database using LOGRETAIN RECOVERY
   ```

   To set `LOGARCHMETH`:

   ```
   db2 update db cfg for database using LOGARCHMETH1 LOGRETAIN
   db2 update db cfg for database using LOGARCHMETH2 OFF
   ```

2. Make a full backup of the database by issuing the following command.

   ```
   db2 backup db database to device
   ```

3. Place the backup in a directory to which DB2 LUW has access rights. If you get the following message, contact your systems administrator:

   ```
   SQL2061N An attempt to access media "device" is denied.
   ```

Specifying the Archive Path

Set the DB2 LUW `OVERFLOWLOGPATH` parameter to the archive log directory. The node attaches automatically to the path variable that you specify.

```
db2 connect to database
db2 update db cfg using overflowlogpath "path"
```

Exclude the node itself from the path. For example, if the full path to the archive log directory is `/sdb2logarch/oltpods1/archive/OLTPODS1/NODE0000`, then the `OVERFLOWLOGPATH` value should be specified as `/sdb2logarch/oltpods1/archive/OLTPODS1`.

Preparing Tables for Processing

The following table attributes must be addressed in an Oracle GoldenGate environment.

- Disabling Triggers and Cascade Constraints
- Ensuring Row Uniqueness for Tables
- Preventing Key Changes
- Enabling Change Capture

- Maintaining Materialized Query Tables

Disabling Triggers and Cascade Constraints

Disable triggers, cascade delete constraints, and cascade update constraints on target Teradata tables. Oracle GoldenGate replicates DML that results from a trigger or cascade constraint. If the same trigger or constraint gets activated on the target table, it becomes redundant because of the replicated version, and the database returns an error. Consider the following example, where the source tables are `emp_src` and `salary_src` and the target tables are `emp_targ` and `salary_targ`.

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

Ensuring Row Uniqueness for Tables

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

Unless a `KEYCOLS` clause is used in the `TABLE` or `MAP` statement, Oracle GoldenGate selects a row identifier to use in the following order of priority:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. If none of the preceding key types exist (even though there might be other types of keys defined on the table) Oracle GoldenGate constructs a pseudo key of all columns that the database allows to be used in a unique key, excluding those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration.

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

4. If a table does not have an appropriate key, or if you prefer that the existing key(s) are not used, you can define a substitute key, if the table has columns that always contain unique values. You define this substitute key by including a KEYCOLS clause within the Extract `TABLE` parameter and the Replicat `MAP` parameter. The specified key will override any

existing primary or unique key that Oracle GoldenGate finds. See TABLE | MAP in *Reference for Oracle GoldenGate*.

- Using KEYCOLS to Specify a Custom Key

Using KEYCOLS to Specify a Custom Key

If a table does not have one of the preceding types of row identifiers, or if you prefer those identifiers not to be used, you can define a substitute key if the table has columns that always contain unique values. You define this substitute key by including a KEYCOLS clause within the Extract TABLE parameter and the Replicat MAP parameter. The specified key will override any existing primary or unique key that Oracle GoldenGate finds.

Preventing Key Changes

Do not add columns to a key after Oracle GoldenGate starts extracting data from the table. This rule applies to a primary key, a unique key, a KEYCOLS key, or an all-column key. DB2 LUW does not supply a before image for columns that are added to a table. If any columns in a key are updated on the source, Oracle GoldenGate needs a before image to compare with the current values in the target table when it replicates the update.

Enabling Change Capture

Configure DB2 to log data changes in the expanded format that is supplied by the DATA CAPTURE CHANGES feature of the CREATE TABLE and ALTER TABLE commands. This format provides Oracle GoldenGate with the entire before and after images of rows that are changed by update statements. You can use GGSCI to issue the ALTER TABLE command as follows.

To Enable Change Capture from GGSCI:

1. From the Oracle GoldenGate directory, run GGSCI.
2. Log on to DB2 from GGSCI as a user that has ALTER TABLE privileges. Specify the data source name with SOURCEDB and specify the user login with USERID and PASSWORD.

   ```
   DBLOGIN SOURCEDB dsn, USERID user[, PASSWORD password]
   ```

3. Issue the following command. where *owner.table* is the fully qualified name of the table. You can use a wildcard to specify multiple table names. Only the asterisk (*) wildcard is supported for DB2 LUW.

   ```
   ADD TRANDATA owner.table
   ```

   ADD TRANDATA issues the following command, which includes logging the before image of LONGVAR columns:

   ```
   ALTER TABLE name DATA CAPTURE CHANGES INCLUDE LONGVAR COLUMNS;
   ```

Example 2-1 To Exclude LONGVAR Logging:

To omit the INCLUDE LONGVAR COLUMNS clause from the ALTER TABLE command, use ADD TRANDATA with the EXCLUDELONG option.

```
ADD TRANDATA owner.table, EXCLUDELONG
```

> **Note:**
>
> If LONGVAR columns are excluded from logging, the Oracle GoldenGate features that require before images, such as the GETUPDATEBEFORES, NOCOMPRESSUPDATES, and NOCOMPRESSDELETES parameters, might return errors if tables contain those columns. For a workaround, see the REQUIRELONGDATACAPTURECHANGES | NOREQUIRELONGDATACAPTURECHANGES options of the TRANLOGOPTIONS parameter.

Maintaining Materialized Query Tables

To maintain parity between source and target materialized query tables (MQT), you replicate the base tables, but not the MQTs. The target database maintains the MQTs based on the changes that Replicat applies to the base tables.

The following are the rules for configuring these tables:

- Include the base tables in your TABLE and MAP statements.
- Do not include MQTs in the TABLE and MAP statements.
- Wildcards can be used in TABLE and MAP statements, even though they might resolve MQT names along with regular table names. Oracle GoldenGate automatically excludes MQTs from wildcarded table lists. However, any MQT that is explicitly listed in an Extract TABLE statement by name will cause Extract to abend.

Database Configuration for DB2 LUW

- The Oracle GoldenGate Extract process calls the DB2READLOG function in the Administrative API to read the transaction log files of a DB2 LUW source database. In addition to DB2READLOG, Extract uses a small number of other API routines to check the source database configuration on startup.
- The Oracle GoldenGate Replicat process uses the DB2 CLI interface on a DB2 LUW target database. For instructions on installing this interface, see the DB2 documentation.
- The database can reside on a different server from the one where Oracle GoldenGate is installed, so long as the database is defined locally. For example, the following enables you to use database mydb locally with data that is on abc123:

  ```
  catalog tcpip node abc123 remote abc123.us.mycompany.com server 00000
  catalog db mydb as abc123 at node abc123 AUTHENTICATION server
  ```

- The DB2 Universal Database has an internal trace facility called db2trc, which acquires Interprocess Communication resources (IPC) (both semaphore and shared memory). Even though a DB2 trace is not turned on, it may issue semget() calls to the operating system. These calls fail since no IPC resources are acquired so you must issue the following command on the DB2 client:

  ```
  db2trc alloc
  ```

- For best performance for DB2 clients with a local database, Oracle recommends that you create a local node catalog instead of TCP/IP when connecting Oracle GoldenGate to a

database that resides on the same machine. This is because local node uses IPC, which is much faster than a TCP/IP node that uses a socket API to access the local database.

- To connect to DB2 LUW remotely from another system, you must use the following driver packages from IBM:
  - IBM Data Server Runtime Client
  - IBM Data Server Driver Package (DS Driver)
  - IBM Data Server Client

  The IBM Data Server Driver for ODBC and CLI (CLI Driver) is not supported for DB2 LUW.

Database User for Oracle GoldenGate Processes for DB2 LUW

- Create a database user that is dedicated to Oracle GoldenGate. It can be the same user for all of the Oracle GoldenGate processes that must connect to a database:
  - Extract (source database)
  - Replicat (target database)
  - `DEFGEN` (source or target database)
- To preserve the security of your data, and to monitor Oracle GoldenGate processing accurately, do not permit other users, applications, or processes to log on as, or operate as, the Oracle GoldenGate database user. It is recommended that you store the login credentials in an Oracle GoldenGate credential store. The credential store makes use of local secure storage for the login names and passwords, and permits you to specify only an alias in the Oracle GoldenGate parameter files.
- Assign system administrator (`SYSADM`) or database administrator (`DBADM`) authority to the database user under which Extract runs. To give the Extract user `DBADM` authority, a user with `SYSADM` authority can issue the following grant statement.

  ```
  GRANT DBADM ON DATABASE TO USER user
  ```

  This authority can also be granted from the `User and Group Objects` folder in the DB2 Control Center. The database tab for the user that is assigned to an Oracle GoldenGate process should have the Database Administrative Authority box checked.

> **Note:**
>
> If the Extract user does not have the required authority, Extract will log the following errors and stop.

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

- Grant at least the following privileges to the database user under which Replicat runs:
  - Local `CONNECT` to the target database
  - `SELECT` on the system catalog views
  - `SELECT`, `INSERT`, `UPDATE`, and `DELETE` on the target tables

Setting the Session Character Set

To support the conversion of character sets between the source and target databases, make certain that the session character set is the same as the database character set. You can set the session character set with the `DB2CODEPAGE` environment variable.

Preparing for Initial Extraction

During the initialization of the Oracle GoldenGate environment, you will be doing an initial data synchronization and starting the Oracle GoldenGate processes for the first time. In conjunction with those procedures, you will be creating process groups. To create an Extract group, an initial start position must be established in the transaction log. This initial read position is on a transaction boundary that is based on one of the following:

- End of the transaction file
- A specific LRI value

The start point is specified with the `BEGIN` option of the `ADD EXTRACT` command.

When the Extract process starts for the first time, it captures all the transaction data that it encounters after the specified start point, but none of the data that occurred *before* that point. This can cause partial transactions to be captured if open transactions span the start point.

To ensure initial transactional consistency:

To avoid the capture of partial transactions, initialize the Extract process at a point in time when the database is in a paused state. DB2 LUW provides a `QUIESCE` command for such a purpose. This is the only way to ensure transactional consistency.

> **Note:**
>
> After the Extract is past the initialization, subsequent restarts of the Extract do not extract partial transactions, because the process uses recovery checkpoints to mark its last read position.

To view open transactions:

IBM provides a utility called `db2pd` for monitoring DB2 databases and instances. You can use it to view information about open transactions and to determine if any of them span the start point. However, because DB2 LUW log records lack timestamps, it might not be possible to make an accurate assessment. If possible, quiesce the database prior to initialization of Oracle GoldenGate.

For more information on initializing the Oracle GoldenGate environment, see Instantiating Oracle GoldenGate with an Initial Load in *Administering Oracle GoldenGate*.

Specifying the DB2 LUW Database in Parameter Files

For an Oracle GoldenGate process to connect to the correct DB2 LUW database, you must specify the name (not an alias) of the DB2 LUW database with the following parameters:

- Specify the DB2 source database with the Extract parameter `SOURCEDB`.
- Specify the DB2 target database name with the Replicat parameter `TARGETDB`.

For more information about these parameters, see the Reference for Oracle GoldenGate for Windows and UNIX.

3
Configuring Oracle GoldenGate for DB2 LUW

This chapter provides an overview of the basic steps required to configure Oracle GoldenGate for a DB2 LUW source and target database.
Topics:

What to Expect from these Instructions

These instructions show you how to configure basic parameter (configuration) files for the following processes:

- A primary Extract (captures transaction data from the data source)
- A data-pump Extract (propagates the data from local storage to the target system)
- A Replicat (applies replicated data to the target database)

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.

Where to Get More Information

See *Administering Oracle GoldenGate* and *Oracle GoldenGate Security Guide* for more information about:

- The processes and files that you are configuring
- Detailed configuration information
- Security options
- Data-integration options (filtering, mapping, conversion)

- Instructions for configuring complex topologies
- Steps to perform initial instantiation of the replication environment

Configuring the Primary Extract

These steps configure the primary Extract to capture transaction data from a source DB2 for i and write the data to a local trail for temporary storage.

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

   ```
   EXTRACT finance
   SOURCEDB mysource, USERIDALIAS myalias
   ENCRYPTTRAIL AES192
   EXTTRAIL /ggs/dirdat/lt
   TABLE hr.*;
   ```

Parameter	Description
`EXTRACT group`	*group* is the name of the Extract group.
`SOURCEDB database, USERIDALIAS alias`	Specifies the real name of the source DB2 for i database (not an alias), plus 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. For more information, see Database User for Oracle GoldenGate Processes.
`ENCRYPTTRAIL algorithm`	Encrypts the local trail.
`EXTTRAIL pathname`	Specifies the path name of the local trail to which the primary Extract writes captured data for temporary storage.

Parameter	Description

`TABLE schema.object;`
`TABLE library/file;`
`TABLE library/file(member);`
	Specifies the database object for which to capture data.
	`TABLE` is a required keyword.
	`schema` is the schema name or a wildcarded set of schemas.
	`object` is the table name, or a wildcarded set of tables.
	`library` is the IBM i library name or a wildcarded set of libraries.
	`file` is the IBM i physical file name or a wildcarded set of physical files.
	`member` is the IBM i physical file member name or a wildcarded set of member names.
	When using the IBM i native name format (library/file with optional member) the only valid wildcards are a name with at least one valid character followed by a trailing asterisk (`*`) or `*ALL` which matches any name.
	
	Note:
	The member name is optional, and must be provided if the member names are required to be written in the trail as part of the object name. Without member names all members in a physical file be implicitly merged as a single object in the trail.
	
	Terminate the parameter statement with a semi-colon.
	To exclude tables from a wildcard specification, use the `TABLEEXCLUDE` parameter.

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.
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 on the target. The data pump is optional, but recommended.

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 Extract 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:

   ```
   EXTRACT extpump
   SOURCEDB mypump, USERIDALIAS myalias
   RMTHOST fin1, MGRPORT 7809 ENCRYPT AES192, KEYNAME securekey2
   RMTTRAIL /ggs/dirdat/rt
   TABLE hr.*;
   ```

Parameter	Description
`EXTRACT group`	*group* is the name of the data pump Extract.
`SOURCEDB database, USERIDALIAS alias`	Specifies the real name of the source DB2 LUW database (not an alias), plus 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 hostname, MGRPORT portnumber, [, ENCRYPT algorithm KEYNAME keyname]`	• `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.
`RMTTRAIL pathname`	Specifies the path name of the remote trail.
`TABLE schema.object;`	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. • `TABLE` is a required keyword. • *schema* is the schema name or a wildcarded set of schemas. • *object* is the name of a table or a wildcarded set of tables. Only the asterisk (*) wildcard is supported for DB2 LUW. The question mark (?) wildcard is not supported for this database. Terminate the parameter statement with a semi-colon. To exclude tables from a wildcard specification, use the `TABLEEXCLUDE` parameter. For more information and for additional `TABLE` options that control data filtering, mapping, and manipulation, see `TABLE | MAP` in *Reference for Oracle GoldenGate*.

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.
4. Save and close the file.

Creating a Temporal Table

A temporal table is a table that maintains the history of its data and the time period when its data are valid. Temporal tables are used in Oracle GoldenGate to keep track of all the old rows that are deleted or updated in the table. Temporal tables are also used to maintain the business validity of its rows and data. For example, Oracle GoldenGate keeps track of the time period during which a row is valid. There are three types of temporal tables, system-period, application-period, and bitemporal table.

- Support for Temporal Tables
- Replicating with Temporal Tables
- Converting
- Creating a Checkpoint Table
- Configuring the Replicat Parameter File

Support for Temporal Tables

- Replication between system-period temporal tables and application-period temporal tables is not supported.
- Replication from a non-temporal table to a temporal table is not supported.
- Replication of temporal tables with the `INSERTALLRECORDS` parameter is not supported.
- Bidirectional replication is supported only with the default replication.
- CDR in bidirectional replication is not supported.
- CDR in application-period temporal tables is supported.

Replicating with Temporal Tables

You can choose one of the following methods to replicate a system-period or a bitemporal temporal table as follows:

- You can replicate a temporal table to another temporal table only; this is the default behavior. Oracle GoldenGate will not replicate the `SYSTEM_TIME` period and transaction id columns because these are automatically generated columns at the apply side. The database manager populates the columns in the target temporal table using the system clock time and with the default values. You can preserve the original values these columns then use any of the following:
  - Add extra timestamp columns in the target temporal table and map the columns accordingly. The extra columns are automatically added to the associated history table.
  - Use a non-temporal table at the apply side and map the columns appropriately. In this scenario, you will not be able to maintain the history table.
  - In a heterogeneous configuration where the source is DB2 LUW and the target is a different database, you can either ignore the automatically generated columns or use an appropriate column conversion function to convert the columns value in the format that target database supports and map them to target columns accordingly.

  Or
- You can replicate a temporal table, with the associated history table, to a temporal and history table respectively then you must specify the replicate parameter, `DBOPTIONS SUPPRESSTEMPORALUPDATES`. You must specify both the temporal table and history table to be captured in the Extract parameter file. Oracle GoldenGate replicates the `SYSTEM_TIME` period and transactions id columns value. You must ensure that the database instance has the execute permission to run the stored procedure at the apply side.

Oracle GoldenGate cannot detect and resolve conflicts while using default replication as `SYSTEM_TIME` period and `transactionstart id` columns remains auto generated. These columns cannot be specified in `set` and `where` clause. If you use the `SUPPRESSTEMPORALUPDATES` parameter, then Oracle GoldenGate supports CDR.

Converting

You can convert an already existing table into a temporal table, which changes the structure of the table. This section describes how the structure of the tables changes. The following

sample existing table is converted into all three temporal tables types in the examples in this section:.

```
Table policy_info
(
Policy_id char[4] not null primary key,
Coverage int not null
             )
And the tables contains the following initial rows
                  POLICY_ID       COVERAGE
                  -------------       -----------
                    ABC               12000
                    DEF                13000
                    ERT                14000
```

Example 1 Converting an existing table into System-period temporal table. You convert the sample existing table into a system-period temporal table by adding `SYSTEM_PERIOD`, `transaction id` columns, and `SYSTEM_TIME` period as in the following:

```
ALTER TABLE policy_info
   ADD COLUMN sys_start TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN;
ALTER TABLE policy_info
   ADD COLUMN sys_end TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END;
ALTER TABLE policy_info
   ADD COLUMN ts_id TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION
START ID;
ALTER TABLE policy_info ADD PERIOD SYSTEM_TIME(sys_start, sys_end);
```

Then you create a history table for the new temporal table using one of the following two methods:

- ```
      CREATE TABLE hist_policy_info
  (
   policy_id     CHAR(4) NOT NULL,
  coverage     INT NOT NULL,
  sys_start    TIMESTAMP(12) NOT NULL ,
  sys_end      TIMESTAMP(12) NOT NULL,
  ts_id            TIMESTAMP(12) NOT NULL
             );
   ALTER TABLE hist_policy_info ADD RESTRICT ON DROP;
  ```
- `CREATE TABLE hist_policy_info LIKE policy_info with RESTRICT ON DROP;`

  The `RESTRICT ON DROP` clause will not allow the history table to get dropped while dropping system-period temporal table. Otherwise the history table gets implicitly dropped while dropping its associated temporal table. You can create a history table without `RESTRICT ON DROP`. A history table cannot be explicitly dropped.

  You should not use the `GENERATED ALWAYS` clause while creating a history table. The primary key of the system-period temporal table also does not apply here as there could be many updates for a particular row in the base table, which triggers many inserts into the history table for the same set of primary keys. Apart from these, the structure of a history table should be exactly same as its associated system-period temporal table. The history table must have the same number and order of columns as system-period temporal table. History table columns cannot
  ```

explicitly be added, dropped, or changed. You must associate a system-period temporal table with its history table with the following statement:

```
ALTER TABLE policy_info ADD VERSIONING USE HISTORY TABLE hist_policy_info.
```

The GENERATED ALWAYS columns of the table are the ones that are always populated by the database manager so you do not have any control over these columns. The database manager populates these columns based on the system time.

The extra added SYSTEM_PERIOD and transaction id columns will have default values for already existing rows as in the following:

```
POLICY_ID                                 COVERAGE
SYS_START
SYS_END                                                  TS_ID
--------- ----------- --------------------------------
--------------------------------
--------------------------------------------------------------------------------
ABC             12000 0001-01-01-00.00.00.000000000000
9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000
DEF             13000 0001-01-01-00.00.00.000000000000
9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000
ERT             14000 0001-01-01-00.00.00.000000000000
9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000
```

The associated history table is populated with the before images once you start updating the temporal table.

Example 2 Converting an existing table into application-period temporal table.
You can convert the sample existing table into application-period temporal table by adding time columns and a BUSINESS_TIME period as in the following:

```
ALTER TABLE policy_info ADD COLUMN bus_start DATE NOT NULL DEFAULT '10/10/2001'"
ALTER TABLE policy_info ADD COLUMN bus_end DATE NOT NULL DEFAULT '10/10/2002'
ALTER TABLE policy_info ADD PERIOD BUSINESS_TIME(bus_start, bus_end)
```

While adding time columns, you need to make sure that while entering business validity time values of the existing time columns, the bus_start column always has value lesser than bus_end because these columns specify the business validity of the rows.
The new application-period temporal table will look similar to:

```
POLICY_ID   COVERAGE    BUS_START  BUS_END
--------- ----------- ---------- ------------------------------
ERT             14000             10/10/2001  10/10/2002
DEF             13000            10/10/2001   10/10/2002
ABC             12000            10/10/2001   10/10/2002
```

Example 3 Converting an existing table into bitemporal table.
You can convert the sample existing table into bitemporal table by adding SYSTEM_PERIOD, time columns along with the SYSTEM_TIME and BUSINESS_TIME period as in the following:

```
ALTER TABLE policy_info
   ADD COLUMN sys_start TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN;
ALTER TABLE policy_info
   ADD COLUMN sys_end TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END;
ALTER TABLE policy_info
   ADD COLUMN ts_id TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID;
ALTER TABLE policy_info ADD PERIOD SYSTEM_TIME(sys_start, sys_end);
```

```
ALTER TABLE policy_info ADD COLUMN bus_start DATE NOT NULL DEFAULT '10/10/2001'"
ALTER TABLE policy_info ADD COLUMN bus_end DATE NOT NULL DEFAULT '10/10/2002'
ALTER TABLE policy_info ADD PERIOD BUSINESS_TIME(bus_start, bus_end)
```

While adding the time columns, you must make sure that while entering business validity time values of already existing time columns, the `bus_start` column always has value lesser than `bus_end` because these columns specify the business validity of the rows.
Then you create a history table for the new temporal table using one of the following two methods:

- ```
      CREATE TABLE hist_policy_info
  (
   policy_id     CHAR(4) NOT NULL,
  coverage     INT NOT NULL,
  sys_start    TIMESTAMP(12) NOT NULL ,
  sys_end      TIMESTAMP(12) NOT NULL,
  ts_id            TIMESTAMP(12) NOT NULL
             );
   ALTER TABLE hist_policy_info ADD RESTRICT ON DROP;
  CREATE TABLE hist_policy_info LIKE policy_info with RESTRICT ON DROP;
  ```
- The `RESTRICT ON DROP` clause will not allow the history table to get dropped while dropping system-period temporal table. Otherwise the history table gets implicitly dropped while dropping its associated temporal table. You can create a history table without `RESTRICT ON DROP`. A history table cannot be explicitly dropped.

  You should not use the `GENERATED ALWAYS` clause while creating a history table. The primary key of the system-period temporal table also does not apply here as there could be many updates for a particular row in the base table, which triggers many inserts into the history table for the same set of primary keys. Apart from these, the structure of a history table should be exactly same as its associated system-period temporal table. The history table must have the same number and order of columns as system-period temporal table. History table columns cannot explicitly be added, dropped, or changed. You must associate a system-period temporal table with its history table with the following statement:

  ```
   ALTER TABLE policy_info ADD VERSIONING USE HISTORY TABLE hist_policy_info.
  ```

  The `GENERATED ALWAYS` columns of the table are the ones that are always populated by the database manager so you do not have any control over these columns. The database manager populates these columns based on the system time.

  The extra added `SYSTEM_PERIOD` and `transaction id` columns will have default values for already existing rows as in the following:

  ```
  POLICY_ID                                 COVERAGE
  SYS_START
  SYS_END                                                 TS_ID
  --------- ----------- --------------------------------
  --------------------------------
  ------------------------------------------------------------------------------
  ---
  ABC             12000 0001-01-01-00.00.00.000000000000
  9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000
  DEF             13000 0001-01-01-00.00.00.000000000000
  ```
  ```

```
9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000
ERT             14000 0001-01-01-00.00.00.000000000000
9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000
```

The associated history table is populated with the before images once you start updating the temporal table.

The extra added `SYSTEM_TIME` period, transaction id and time columns will have default values for already existing rows as in the following:

```
POLICY_ID COVERAGE    SYS_START
SYS_END                         TS_ID                           BUS_START
BUS_END
--------- ----------- ------------------------------
------------------------------ ------------------------------ ----------
------------------------------------
ABC   12000 0001-01-01-00.00.00.000000000000 9999-12-30-00.00.00.000000000000
0001-01-01-00.00.00.000000000000 10/10/2001 10/10/2002
DEF             13000 0001-01-01-00.00.00.000000000000
9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000 10/10/2001
10/10/2002
ERT             14000 0001-01-01-00.00.00.000000000000
9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000 10/10/2001
10/10/2002
```

The history table is populated with the before images once user starts updating the temporal table.

Example 4 Replication in Heterogeneous Environment.
In heterogeneous configuration in which you do not have temporal tables at the apply side, you can only replicate the system-period and bitemporal tables though *not* the associated history tables. While performing replication in this situation, you must take care of the `SYSTEM_PERIOD` and transaction id columns value. These columns will have some values that the target database might not support. You should first use the map conversion functions to convert these values into the format that the target database supports, and then map the columns accordingly.
For example, MySQL has a `DATETIME` range from `1000-01-01 00:00:00.000000` to `9999-12-31 23:59:59.999999`. You cannot replicate a timestamp value of `0001-01-01-00.00.00.000000000000` to MySQL. To replicate such values, you must convert this value into the MySQL `DATETIME` value `1000-01-01 00:00:00.000000`, and then map the columns. If you have the following row in the `policy_info` system-period table:

```
POLICY_ID                                COVERAGE
SYS_START
SYS_END                                                TS_ID
--------- ----------- ------------------------------
------------------------------
--------------------------------------------------------------------------------
ABC             12000 0001-01-01-00.00.00.000000000000
9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000
```

To replicate the row into MySQL, you would use the `colmap()` function:

```
map source_schema.policy_info, target target_schema.policy_info colmap
(policy_id=policy_id, coverage=coverage, sys_start= @IF( ( @NUMSTR( @STREXT(sys_
start,1,4))) > 1000, sys_start, '1000-01-01 00.00.00.000000'), sys_end=sys_end,
 ts_id= @IF( ( @NUMSTR( @STREXT(ts_id,1,4))) > 1000, ts_id, '1000-01-01
 00.00.00.000000'));
```

Creating a Checkpoint Table

The checkpoint table is a required component of Replicat.

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.

To configure a checkpoint table, see Creating a Checkpoint Table in *Administering Oracle GoldenGate*.

Configuring the Replicat Parameter File

These steps configure the Replicat process. This process applies replicated data to a DB2 LUW target database.

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.

   Basic parameters for the Replicat group:

   ```
   REPLICAT financer
   TARGETDB mytarget, USERIDALIAS myalias
   ASSUMETARGETDEFS
   MAP hr.*, TARGET hr2.*;
   ```

Parameter	Description
`REPLICAT group`	*group* is the name of the Replicat group.
`TARGETDB database, USERIDALIAS alias`	Specifies the real name of the target DB2 LUW database (not an alias), plus 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 Database User for Oracle GoldenGate Processes.
`ASSUMETARGETDEFS`	Specifies how to interpret data definitions. `ASSUMETARGETDEFS` assumes the source and target tables have identical definitions. (This procedure assume identical definitions.) Use the alternative `SOURCEDEFS` if the source and target tables have different definitions, and create a source data-definitions file with the `DEFGEN` utility.

Parameter	Description
`MAP` *`schema.object`*`,` `TARGET` *`schema.object`*`;`	Specifies the relationship between a source table or multiple objects, and the corresponding target object or objects. • `MAP` specifies the source portion of the `MAP` statement and is a required keyword. Specify the source objects in this clause. • `TARGET` specifies the target portion of the `MAP` statement and is a required keyword. Specify the target objects to which you are mapping the source objects. • *`schema`* is the schema name or a wildcarded set of schemas. • *`object`* is the name of a table or a wildcarded set of objects. Terminate this parameter statement with a semi-colon. Note that only the asterisk (*) wildcard is supported for DB2 LUW. The question mark (?) wildcard is not supported for this database. To exclude objects from a wildcard specification, use the `MAPEXCLUDE` parameter.

3. 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.
4. Save and close the file.

When to Start Replicating Transactional Changes

You must start replication when the source and target data is in a synchronized state, where the corresponding rows in the source and target tables contain identical data values. Unless you are starting with brand new source and target databases with no current user activity, you will need to activate change capture and apply processes to handle ongoing transactional changes while an initial load is being applied to the target. This process is known as *initial synchronization*, or also as *instantiation*. The initial load captures a point-in-time snapshot of the source data and applies it to the target, while Oracle GoldenGate maintains any changes that are made after that point.

See Instantiating Oracle GoldenGate with an Initial Load in *Administering Oracle GoldenGate* for instantiation options.

Testing Your Configuration

It is important to test your configuration in a test environment before deploying it 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.

Part II
Using Oracle GoldenGate for DB2 for i

Oracle GoldenGate for DB2 for i supports capture and delivery of initial load and transactional data for supported DB2 for i database versions.

Oracle GoldenGate for DB for i supports the mapping, filtering, and transformation of source data, unless noted otherwise in this document, along with replicating data derived from other source databases supported by Oracle GoldenGate, into DB2 for i databases.

This part describes tasks for configuring and running Oracle GoldenGate for DB2 for i.

Topics:

4

Understanding What's Supported for DB2 for i

This chapter contains information on database and table features supported by Oracle GoldenGate for DB2 for i.

Oracle GoldenGate on DB2 for i supports the filtering, mapping, and transformation of data unless otherwise noted in this documentation.

Oracle GoldenGate for DB2 for i runs remotely from a Linux system on a DB2 for i source system to capture data from the transaction journals for replication to a target system. To apply data to a target DB2 for i database, Oracle GoldenGate can run remotely from a Linux system. Oracle GoldenGate communicates with the IBM i system by means of an ODBC connection, and no Oracle GoldenGate software is installed on the DB2 for i target.

> **Note:**
>
> The DB2 for i platform uses one or more **journals** to keep a record of transaction change data. For consistency of terminology in the supporting administrative and reference Oracle GoldenGate documentation, the terms "log" or "transaction log" may be used interchangeably with the term "journal" where the use of the term "journal" is not explicitly required.

Topics:

Supported DB2 for i Data Types

Oracle GoldenGate supports all DB2 for i data types, except those listed in Non-Supported DB2 for i Data Types.

Limitations of support

Extract fully supports the capture and apply of `TIMESTAMP(0)` through `TIMESTAMP(12)` when the output trail format is 19.1 or higher. Otherwise Extract truncates data from `TIMESTAMP(10)`

through `TIMESTAMP(12)` to nanoseconds (maximum of nine digits of fractional time) and issues a warning to the report file.

Oracle GoldenGate supports timestamp data from 0001/01/03:00:00:00.000000 to 9999/12/31:23:59:59.999999. If a timestamp is converted from GMT to local time, these limits also apply to the resulting timestamp. Depending on the time zone, conversion may add or subtract hours, which can cause the timestamp to exceed the lower or upper supported limit.

Non-Supported DB2 for i Data Types

Oracle GoldenGate does not support the following DB2 for i data types:

- `XML`
- `DATALINK`
- User-defined types

Supported Objects and Operations for DB2 for i

Oracle GoldenGate supports the following DB2 for i objects and operations.

- Extraction and replication of DML operations .
- Tables with the maximum row length supported by the database.
- Tables that contain up to the maximum number of columns that is supported by the database, up to the maximum supported column size.
- `TRUNCATE` operations are supported and are represented by `DELETE FROM` with no `WHERE` clause SQL statements and `Clear Physical File Member` (`CLRPFM`).
- Base tables underlying Materialized Query Tables, but not the MQTs themselves. The target database automatically maintains the content of the MQT based on the changes that are applied to the base tables by Replicat.
- Both Library (Native) names including members, and SQL names are allowed.
- Partitioned tables
- Supported options with `SHOWTRANS`:

```
SHOWTRANS [transaction_ID]  [COUNT n]
[DURATION duration unit] [TABULAR]
[FILE file_name] |
```

- Options for `SKIPTRANS` and `FORCETRANS`:

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

- Limitations on Automatic Heartbeat Table support are as follows:
  - The `ADD HEARTBEATTABLE` command creates a new file called `ogghbfreq` in the Oracle GoldenGate installation directory. Do not delete this file because the `pase` heartbeat program reads the frequency values from it.

- There is an extra executable in the Oracle GoldenGate build folder named `ogghb`.
- An extra process named `ogghb` starts running on the IBM i system when the `ADD HEARTBEATTABLE` command runs until you disable the heartbeat with the `DELETE HEARTBEATTABLE` command. This process automatically restarts even if it is killed. To remove this process from the system, use the `DELETE HEARTBEATTABLE` command.
- When using the `ALTER HEARTBEATTABLE` command to change the heartbeat frequency with the `PURGE_FREQUENCY` or `RETENTION_TIME` options, it takes approximately 60 + older 'frequency') seconds to be implemented.
- There is an initial delay of 30 seconds between `ADD HEARTBEATTABLE` and the first record is updated in the heartbeat seed table.
- `[THREAD n]` and `[DETAIL]` is not supported.

Non-Supported Objects and Operations for DB2 for i

Oracle GoldenGate does not support the following objects or operations for DB2 for i.

- DDL operations
- Schema, table or column names that have trailing spaces
- Multiple instances of a database

Oracle GoldenGate Parameters Not Supported for DB2 for i

This section lists some of the Oracle GoldenGate configuration parameters that are not supported for the DB2 for i platform. For full descriptions of Oracle GoldenGate parameters and the databases they support, see Oracle GoldenGate Parameters.

`BATCHSQL`
`BR`
`ASCIITOEBCDIC` and `EBCDICTOASCII`
`BINARYCHARS`
`LOBMEMORY`
`TRAILCHARSETEBCDIC`

Supported Object Naming Conventions

Oracle GoldenGate supports SQL naming conventions and also supports native file system names in the format of *`library/file(member)`*.

For native (system) names, Oracle GoldenGate supports the normal DB2 for i naming rules for wildcarding, which allows `*ALL` or a partial name with a trailing asterisk (*) wildcard. For example:

- *`library`*`/*all(*all)`
- *`library`*`/a*(a*)`
- *`library`*`/abcde*`

The member name is optional and may be left off. In that case, data for all of the members will be extracted, but only the library and file names will be captured and included in the records that are written to the trail. The result is that the data will appear to have come from

only one member on the source, and you should be aware that this could cause integrity conflicts on the target if there are duplicate keys across members. To include the member name in the trail records, include the member explicitly or though a wildcarded member specification.

For SQL names, only the first member in the underlying native file is extracted in accordance with the normal operation of SQL on an DB2 for i system. For SQL names, Oracle GoldenGate supports the wildcarding of both table names and schema names. For instructions on wildcarding SQL names, see Specifying Object Names in Oracle GoldenGate Input in *Administering Oracle GoldenGate*.

Port Requirements for DB2 for i

Oracle GoldenGate for Db2 for i requires the following ports in normal and SSL modes:

- **Normal Mode**: 446, 449, 8470-8476
- **SSL Mode**: 448, 449, 9470-9476

System Schemas for DB2 for i

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

- `"Q*"`
- `"SYSIBM"`
- `"SYSIBMADM"`
- `"SYSPROC"`
- `"SYSTOOLS"`
- `"#LIBRARY"`
- `"#RPGLIB"`

Supported Character Sets

The default behavior of a DB2 for i Extract is to convert all character data to Unicode. The overhead of the performance of the conversion to UTF-8 for the text data has been substantially reduced. However, if you want to send data in its native character set you can use the parameter `DBOPTIONS USEDATABASEENCODING` to override the default behavior.

5
Preparing the System for Oracle GoldenGate

This chapter contains guidelines for preparing the DB2 for i system to support Oracle GoldenGate.
Topics:

Preparing the Journals for Data Capture by Extract

All tables for which you want data to be captured must be journaled, either explicitly or by default by means of a `QSQJRN` journal in the same library. To preserve data integrity, data journal entries are sent to the Extract process in time order as they appear on the system. This section provides guidelines for configuring the journals to support capture by the Extract process.

Allocating Journals to an Extract Group

One Extract process can process a single journal. If using more journals than that, use additional Extract processes to handle the extra journals. You can also use additional Extract processes to improve capture performance if necessary.

> **Note:**
>
> To ensure transaction integrity, all objects that correspond to any given transaction must be read by the same Extract group. For more information about using multiple Extract processes, see Tuning the Performance of Oracle GoldenGate in *Administering Oracle GoldenGate*.

Setting Journal Parameters

To support the capture of data by the Extract process, the following are the minimal journaling parameter settings that are required.

- Manage Receivers `(MNGRCV): *SYSTEM`
- Delete Receivers `(DLTRCV): *NO`

- Receiver Size Option (RCVSIZOPT): *MAXOPT2 (*MAXOPT3 recommended to avoid the necessity to perform an ALTER EXTRACT with the ETROLLOVER option when the journal sequence numbers run out if *MAXOPT2 is used.)
- Journal State (JRNSTATE): *ACTIVE
- Minimize Entry Specific Data (MINENTDTA): *NONE
- Fixed Length Data (FIXLENDTA): *USR

In the following example, the command to set these attributes for a journal JRN1 in library LIB1 would be:

```
CHGJRN JRN(LIB1/JRN1) MNGRCV(*SYSTEM) DLTRCV(*NO) RCVSIZOPT(*MAXOPT3)
JRNSTATE(*ACTIVE) MINENTDTA(*NONE) FIXLENDTA(*USR)
```

> **Note:**
>
> To check the attributes of a journal, use the command WRKJRNA JRN(LIB1/JRN1) DETAIL(*CURATR).

When the journaling is set to the recommended parameter settings, you are assured that the entries in the journals contain all of the information necessary for Oracle GoldenGate processing to occur. These settings also ensure that the system does not delete the journal receivers automatically, but retains them in case Extract needs to process older data.

Deleting Old Journal Receivers

Although the DLTRCV parameter is set to NO in the recommended configuration for Extract (see Setting Journal Parameters), you can delete old journal receivers manually once Extract is finished capturing from them.

If using another application that is using the journals that Oracle GoldenGate will be reading, consideration must be given regarding any automatic journal receiver cleanup that may be in place. Oracle GoldenGate must be able to read the journal receivers before they are detached or removed.

To Delete Journal Receivers

1. Run GGSCI.
2. In GGSCI, issue the following command to view the journal positions in which Extract has current processing points, along with their journal receivers.

   ```
   INFO EXTRACT group
   ```

3. Use the following DB2 for i command to delete any journal receivers that were generated prior to the ones that are shown in the INFO EXTRACT command.

   ```
   DLTJRNRCV JRNRCV(library/journal_receiver)
   ```

   Where:

library and *journal_receiver* are the actual names of the library and journal receiver to be deleted. See the DB2 for i Information Center for more information about this command.

Specifying Object Names

Oracle GoldenGate commands and parameters support input in the form of SQL names, native names in the format of *library_name/file_name(member_name)*, or a mix of the two. If a native file system name does not include the member name, all members are implicitly selected by the Oracle GoldenGate process. For a SQL name only the first member is used.

To support case sensitivity of double quoted object names, specify those names within double quotes in the Oracle GoldenGate parameter files. This is true of both SQL and native file system names.

When specifying a native table name in a `MAP` statement on a platform other than DB2 for i, the name must be enclosed within double quotes so that Oracle GoldenGate correctly interprets it as a separator character.

For consistency of terminology in other administrative and reference Oracle GoldenGate documentation, the SQL terms "schema" and "table" are used to reference the containers for the DB2 for i data, as shown here.

Table 5-1 Native-SQL object name relationships

Native	SQL	Notes
Library (maximum length 10)	Schema (maximum length 128)	The operating system creates a corresponding native name for a SQL-created schema.
File (maximum length 10)	Table (maximum length 128)	The operating system creates a corresponding native name for a SQL-created table.
Member	Not Applicable	Contains the actual data. Only the first member of a `FILE` object can be accessed through SQL. To access data in other members the native system name must be used.

Preparing Tables for Processing

The following table attributes must be addressed in an Oracle GoldenGate environment.

- Ensuring Row Uniqueness for Tables
- Preventing Key Changes
- Disabling Constraints on the Target
- Enabling Change Capture
- Maintaining Materialized Query Tables

Ensuring Row Uniqueness for Tables

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

Unless a `KEYCOLS` clause is used in the `TABLE` or `MAP` statement, Oracle GoldenGate selects a row identifier to use in the following order of priority:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. If none of the preceding key types exist (even though there might be other types of keys defined on the table) Oracle GoldenGate constructs a pseudo key of all columns that the database allows to be used in a unique key, excluding those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration.

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

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

- Using KEYCOLS to Specify a Custom Key

Using `KEYCOLS` to Specify a Custom Key

If a table does not have one of the preceding types of row identifiers, or if you prefer those identifiers not to be used, you can define a substitute key if the table has columns that always contain unique values. You define this substitute key by including a `KEYCOLS` clause within the Extract `TABLE` parameter and the Replicat `MAP` parameter. The specified key will override any existing primary or unique key that Oracle GoldenGate finds.

Preventing Key Changes

If you must add columns to the key that Extract is using as the row identifier for a table (primary key, unique key, `KEYCOLS` key, or all-column key) after Oracle GoldenGate has started processing journal data, follow these steps to make the change.

1. Stop Extract.

```
STOP EXTRACT group
```

2. Issue the following command until it returns `EOF`, indicating that it has processed all of the existing journal data.

   ```
   INFO EXTRACT group
   ```

3. Make the change to the key.
4. Start Extract.

   ```
   START EXTRACT group
   ```

Disabling Constraints on the Target

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

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

Enabling Change Capture

To capture changes to a table in a journal, you can run the `STRJRNPF` command on the OS/400 command line or run the `ADD TRANDATA` command from GGSCI. The `ADD TRANDATA` command calls `STRJRNPF` and is the recommended method to start journaling for tables, because it ensures that the required journal image attribute of Record Images `(IMAGES)`: `*BOTH` is set on the `STRJRNPF` command.

To Run `ADD TRANDATA`

1. Run GGSCI on the source system.
2. Issue the `DBLOGIN` command.

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

   Where: `SOURCEDB` specifies the default DB 2 for i database, `USERID` specifies the Extract user profile, and `PASSWORD` specifies that profile's password.

3. Issue the `ADD TRANDATA` command.

   ```
   ADD TRANDATA table_specification
   ```

Where: *table_specification* is one of the following:

- *schema.table* [JOURNAL *library/journal*]
- *library/file* [JOURNAL *library/journal*] (See Specifying a Default Journal)

- Specifying a Default Journal
- Removing a Default Journal Specification

Specifying a Default Journal

To specify a default journal for multiple tables or files in the ADD TRANDATA command, instead of specifying the JOURNAL keyword, use the following GGSCI command before issuing ADD TRANDATA.

```
DEFAULTJOURNAL library/journal
```

Any ADD TRANDATA command used without a journal assumes the journal from DEFAULTJOURNAL.

To display the current setting of DEFAULTJOURNAL, you can issue the command with no arguments.

Removing a Default Journal Specification

To remove the use of a default journal, use the following GGSCI command:

```
DEFAULTJOURNAL CLEAR
```

Maintaining Materialized Query Tables

To maintain parity between source and target materialized query tables (MQT), you replicate the base tables, but not the MQTs. The target database maintains the MQTs based on the changes that Replicat applies to the base tables.

The following are the rules for configuring these tables:

- Include the base tables in your TABLE and MAP statements.
- Do not include MQTs in the TABLE and MAP statements.
- Wildcards can be used in TABLE and MAP statements, even though they might resolve MQT names along with regular table names. Oracle GoldenGate automatically excludes MQTs from wildcarded table lists. However, any MQT that is explicitly listed in an Extract TABLE statement by name will cause Extract to abend.

Adjusting the System Clock

It is recommended that you set the system clock to UTC (Universal Time Coordinate) time and use the timezone offset in the DB2 for i system values to represent the correct local time. If this setup is done correctly, local daylight savings time adjustments can occur automatically with no disruption to replication.

6

Configuring Oracle GoldenGate for DB2 for i

This chapter contains instructions for configuring Oracle GoldenGate to capture source DB2 for i data and apply it to a supported target database.

Topics:

What to Expect from this Procedure

These instructions show you how to configure a set of basic Oracle GoldenGate parameter (configuration) files, one for each process that replicates transactional data changes from a DB2 for i source to a DB2 for i target, or to a different database type. Your business requirements probably will require a more complex topology, but this procedure forms a basis for the rest of your configuration steps.

This chapter focuses on the basic parameters that are specific to DB2 for i.

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 additional parameter files faster than starting from scratch.

Getting Started with Oracle GoldenGate

Before proceeding with the configuration process, you should get familiar with the Oracle GoldenGate architecture, the command interface, and the methods for supplying input and instructions to the processes. See *Administering Oracle GoldenGate* for this information.

Creating the Oracle GoldenGate Instance

Each Oracle GoldenGate installation is rooted in the Manager process. This is the controller process that instantiates the Oracle GoldenGate processes, allocates port numbers, and performs file maintenance. Together, the Manager process and its child processes, and their related programs and files comprise an Oracle GoldenGate instance.

To run Oracle GoldenGate, a Manager process must be running on all systems that will be part of the Oracle GoldenGate environment. To run Manager, you first create a parameter file for it.

Creating a GLOBALS File

The GLOBALS parameter file contains parameters that affect all processes within an Oracle GoldenGate instance.

GGSCHEMA is a mandatory parameter for Oracle GoldenGate 21c (21.3.0) onwards and defines the schema, which Oracle GoldenGate uses on the remote system for necessary Oracle GoldenGate database objects.

The GLOBALS parameter `NAMECCSID` is specific to DB2 for i and may be required, if the SQL catalog contains object names that are referenced by a different CCSID than the system CCSID. The SQL catalog is created in the system CCSID and does not indicate this difference when queried. Oracle GoldenGate makes queries to the catalog and could retrieve the name incorrectly unless `NAMECCSID` is used to supply the correct CCSID value. For more information, see *Reference for Oracle GoldenGate*.

Creating a Data Definitions File

When replicating data from one table to another, an important consideration is whether the column structures (metadata) of the source and target tables are identical. Oracle GoldenGate looks up metadata for the following purposes:

> **Note:**
>
> This is only required when writing trails for Oracle GoldenGate 11.2 or earlier.

- On the source, to supply complete information about captured operations to the Replicat process.
- On the target, to determine the structures of the target tables, so that the replicated data is correctly mapped and converted (if needed) by Replicat.

When source and target table definitions are dissimilar, Oracle GoldenGate must perform a conversion from one format to the other. To perform conversions, both sets of definitions must be known to Oracle GoldenGate. Oracle GoldenGate can query the local database to get one set of definitions, but it must rely on a *data-definitions file* to get definitions from the remote database. The data-definitions file contains information about the metadata of the data that is being replicated.

To create a definitions file, you configure and run the `DEFGEN` utility and then transfer the definitions file to the target system. This file must be in place on the target system before you start the Oracle GoldenGate processes for the first time.

Enabling SSL

SSL connections can be enabled by setting `SSL=1` in the DSN configuration file. To know about how to set up an SSL connection with IBM i Access ODBC Driver, see Make SSL ODBC connections from Linux to Db2 for i and ACS ODBC driver for Linux now supports OpenSSL.

It is recommended to use OpenSSL to setup SSL. After SSL is enabled in the DSN configuration file, JAVA connections that have been established using `jt400` libraries will also be using SecureAS400 connections as well.

User Profiles and Security Privileges

The user who installs Oracle GoldenGate must have read and write privileges on the Oracle GoldenGate installation directory, as these privileges will be required later to perform steps to create sub-folders and run specific programs.

In addition, this user must have the following rights to ensure smooth installation of Oracle GoldenGate on Db2 for i:

- The`*ALLOBJ` special authority (required only for installation/upgrade or `HEARTBEAT ADD/DELETE` commands)

> **Note:**
>
> For any new Oracle GoldenGate installation or an upgrade to the latest release version, the user or group profile to be used for the installation, needs to run the `DBLOGIN` command from GGSCI just once to complete the installation of the new objects for the Oracle Golden Gate installation on the IBM i system.

> **Note:**
>
> The objects in the Oracle GoldenGate library (specified with GGSCHEMA), should have their ownership changed to the dedicated user profile for Oracle GoldenGate. If the user does not have the `*ALLOBJ` authority, then any other user with the `*ALLOBJ` authority needs to sign-in to the Db2 for i system, and change the ownership of the objects to the user or group profile to be used for Oracle GoldenGate installation.

- Authority to the `RSTOBJ` command (typically available with `*ALLOBJ`)
- Ability to create a library, if required

Dedicated User Profile Account

It is recommended that the Oracle GoldenGate processes on Db2 for i database be assigned a dedicated user or group profile, and is used by all Oracle GoldenGate processes. This user profile should not be used by any other application(s).

The dedicated user profile should be granted permission only to the objects that the Oracle GoldenGate will be operating on. If there is specific change data that is not to be accessed by Oracle GoldenGate processes, then such change data should not be included in the journals, which are accessed by Oracle GoldenGate and its dedicated user profile. All Oracle GoldenGate processes must have read, write, and delete object privileges within the Oracle GoldenGate installation library, as specified by GGSCHEMA.

Security Privileges on a Db2 for i System

The Extract and Replicat user profiles need to be assigned the following authorities at a minimum:

- The simplest way to ensure Oracle GoldenGate will be able to operate is to assign `*ALLOBJ` authority to the Oracle GoldenGate user profile(s), however this is not necessary.
- The Manager process must have privileges to control all other Oracle GoldenGate processes (DB2 for i `*JOBCTL` authority).
- The Oracle GoldenGate user profiles(s) need at least the `*USE` authority to the`*FILE` objects in the `QSYS2` library which contains the SQL catalog (which by default should be accessible to any user).
- Assign at least the `*USE` authority (`*OBJOPR`, `*READ`, `*EXECUTE`) to all the `*FILE` (table) and `*JRNRCV` (journal receiver) objects on the system that are accessed by the Extract user profile.
- Assign the following authorities to the `*JRN` (journal) objects that are accessed by the Extract user profile, in addition to the `*USE` authority (`*OBJOPR`, `*READ`, `*EXECUTE`): `*OBJEXIST`,`*OBJREF`, and `*ADD`.
- Assign the `*CHANGE` authority to all the `*FILE` objects on the system that are accessed by the Replicat user profile.

The Oracle GoldenGate user profile that runs the Extract process needs to have the `*USE` authority on the `QSYS/QPMLPMGT` service program.

These authorities must be granted through the native DB2 for i interface through a 5250 terminal session or through the DB2 for i Navigator product available from IBM.

Encrypting the Extract and Replicat Passwords

It is strongly recommended that you encrypt the passwords of the user profiles that will be used for the primary and data pump Extracts, and for the Replicat process. The standard Oracle GoldenGate encryption method of AES (Advanced Encryption Standard) is supported by the IBM i platform. To encrypt the password, see Working with Runtime Parameters in *Administering Oracle GoldenGate*. It also contains information about how to encrypt data within disk storage and across TCP/IP.

> **Note:**
>
> The Oracle GoldenGate credential store is not supported by the iSeries platform.

Configuring Extract for Change Capture from DB2 for i

Perform these steps on the source system to configure the primary Extract and the data pump Extract that support change capture and transport across the network.

- Configuring the Primary Extract
- Configuring the Data Pump

Configuring the Primary Extract

These steps configure the primary Extract to capture transaction data from a source DB2 for i and write the data to a local trail for temporary storage.

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

   ```
   EXTRACT finance
   SOURCEDB mysource, USERIDALIAS myalias
   ENCRYPTTRAIL AES192
   EXTTRAIL /ggs/dirdat/lt
   TABLE hr.*;
   ```

Parameter	Description
`EXTRACT group`	*group* is the name of the Extract group.
`SOURCEDB database, USERIDALIAS alias`	Specifies the real name of the source DB2 for i database (not an alias), plus 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. For more information, see Database User for Oracle GoldenGate Processes.
`ENCRYPTTRAIL algorithm`	Encrypts the local trail.
`EXTTRAIL pathname`	Specifies the path name of the local trail to which the primary Extract writes captured data for temporary storage.

Parameter	Description

`TABLE schema.object;`
`TABLE library/file;`
`TABLE library/file(member);`
	Specifies the database object for which to capture data.
	`TABLE` is a required keyword.
	`schema` is the schema name or a wildcarded set of schemas.
	`object` is the table name, or a wildcarded set of tables.
	`library` is the IBM i library name or a wildcarded set of libraries.
	`file` is the IBM i physical file name or a wildcarded set of physical files.
	`member` is the IBM i physical file member name or a wildcarded set of member names.
	When using the IBM i native name format (library/file with optional member) the only valid wildcards are a name with at least one valid character followed by a trailing asterisk (`*`) or `*ALL` which matches any name.
	Note:
	The member name is optional, and must be provided if the member names are required to be written in the trail as part of the object name. Without member names all members in a physical file be implicitly merged as a single object in the trail.
	Terminate the parameter statement with a semi-colon.
	To exclude tables from a wildcard specification, use the `TABLEEXCLUDE` parameter.

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.
4. Save and close the file.

Configuring the Data Pump

These steps configure the data pump that reads the local trail and sends the data across the network to a remote trail.

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 group.

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 the following table for description.

   Basic parameters for the data-pump Extract group:

   ```
   EXTRACT extpump
   SOURCEDB FINANCE USERID ogg, PASSWORD AACAAAAAAAAAAA, BLOWFISH
   ENCRYPTKEY mykey
   RMTHOST fin1, MGRPORT 7809
   ```

```
RMTTRAIL /ggs/dirdat/rt
TABLE hr.*;
```

Parameter	Description
`EXTRACT group`	*group name* is the name of the data pump.
`SOURCEDB database USERID user, PASSWORD password, BLOWFISH ENCRYPTKEY keyname`	Specifies database connection information. • `SOURCEDB` specifies the *default* DB 2 for i database. • `USERID` specifies the Extract database user profile. • `PASSWORD` specifies the user's password that was encrypted with the `ENCRYPT PASSWORD` command. Enter or paste the encrypted password after the `PASSWORD` keyword. • `BLOWFISH ENCRYPTKEY keyname` specifies the name of the lookup key in the local `ENCKEYS` file.
`DECRYPTTRAIL BLOWFISH`	Decrypts the input trail.
`RMTHOST hostname, MGRPORT portnumber`	• `RMTHOST` specifies the name or IP address of the target system. • `MGRPORT` specifies the port number where Manager is running on the target.
`ENCRYPTTRAIL BLOWFISH`	Encrypts the remote trail with Blowfish encryption.
`RMTTRAIL pathname`	Specifies the path name of the remote trail.
`TABLE schema.object; TABLE library/file; TABLE library/file(member);`	Specifies a table or tables to process. Terminate the `TABLE` statement with a semi-colon. To exclude tables from a wildcard specification, use the `TABLEEXCLUDE` parameter after the `TABLE` statement.

3. Enter any optional Extract parameters that are recommended elsewhere in this manual and any others shown in Summary of Extract Parameters in *Reference for Oracle GoldenGate*.
4. Save and close the file.

Configuring Replicat for Change Delivery to DB2 for i

These steps configure Replicat to apply data to a DB2 for i target database, operating on a remote Linux system. To configure Replicat for change delivery to a different database type,

such as an Oracle database, follow the directions in the Oracle GoldenGate Installation and Configuration guide for that database. There may be additional parameters and requirements for delivery to that database type.

> **Note:**
>
> There does not have to be a database on a Windows or Linux machine to support connection by ODBC by Replicat.

Creating a Checkpoint Table

Replicat maintains its checkpoints in a checkpoint table in the DB2 for i target database. Each checkpoint is written to the checkpoint table, that must be journaled, 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.

A common method of create the checkpoint table with journaling is as follows:

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

   Set the name of the journal that is intended to be used for the checkpoint file. The default will be `<GGSCHEMA>/OGGJRN`. You can change it by setting the default for the current GGSCI session using the `DEFAULTJOURNAL` command. The syntax of the `DEFAULTJOURNAL` command is:

   ```
   DEFAULTJOURNAL library_name/journal_name
   ```

   Where: *`library_name`* is the name of the library and `journal_name` is the name of the journal to be used for subsequent operations that may optionally have a journal specified..

2. Add the checkpoint table.

   ```
   ADD CHECKPOINTTABLE library_name.chkptab

   Successfully created checkpoint table kgr.chkptab
   ```

3. Add journaling to the checkpoint table.

   ```
   ADD TRANDATA library_name.CHKPTAB
   ```

For more information about creating a checkpoint table, see *Administering Oracle GoldenGate*.

Configuring Replicat

These steps configure the Replicat process in a basic way without any special mapping or conversion of the data.

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.

   ```
   REPLICAT financer
   TARGETDB FINANCIAL USERID ogg, PASSWORD AACAAAAAAAAAAA, BLOWFISH
   ENCRYPTKEY mykey
   DISCARDFILE /users/ogg/disc
   MAP hr.*, TARGET hr2.*;
   ```

Parameter	Description
`REPLICAT group`	*group* is the name of the Replicat group.
`TARGETDB database USERID user, PASSWORD password, BLOWFISH ENCRYPTKEY keyname`	Specifies database connection information. • `SOURCEDB` specifies the data source name (DSN) of the target DB2 for i database. • `USERID` specifies the Replicat database user profile. • `PASSWORD` specifies the user's password that was encrypted with the `ENCRYPT PASSWORD` command. Enter or paste the encrypted password after the `PASSWORD` keyword. • `BLOWFISH ENCRYPTKEY keyname` specifies the name of the lookup key in the local `ENCKEYS` file.
`DECRYPTTRAIL BLOWFISH`	Decrypts the input trail.

Parameter	Description
`MAP owner.table, TARGET owner.table; MAP owner.table, TARGET library/file; MAP library/file, TARGET owner.table; MAP library/file, TARGET library/file;`	Specifies a relationship between a source and target table or tables. The `MAP` clause specifies the source objects, and the `TARGET` clause specifies the target objects to which the source objects are mapped. • *owner* is the schema name. • *table* is the name of a table or a wildcard definition for multiple tables. • `library` is the IBM i library name or a wildcard definition for multiple libraries. • `file` is the IBM i physical file name or a wildcard definition for multiple physical files. **Note:** There is an optional physical file member name also allowed with the physical file of the form `file(member)` and member may also be a wildcard definition for multiple members. To exclude tables from a wildcard specification, use the `MAPEXCLUDE` parameter. For more information and for additional options that control data filtering, mapping, and manipulation, see `MAP` in *Reference for Oracle GoldenGate*.

3. Enter any optional Extract parameters that are recommended elsewhere in this manual and any others shown in Summary of Extract Parameters.
4. Save and close the file.

Next Steps in the Deployment

Because of its flexibility, Oracle GoldenGate offers numerous features and options that must be considered before you start any processes. To further configure Oracle GoldenGate to suit your business needs, see the following:

- For additional configuration guidelines to achieve specific replication topologies, see *Administering Oracle GoldenGate*. This guide also contains information about:
  - Oracle GoldenGate architecture
  - Oracle GoldenGate commands
  - Oracle GoldenGate initial load methods
  - Using customization features
  - Mapping columns that contain dissimilar data
  - Data filtering and manipulation
- For syntax options and descriptions of Oracle GoldenGate GGSCI commands and Oracle GoldenGate parameters shown in this guide, see *Reference for Oracle GoldenGate*.

When to Start Replicating Transactional Changes

You must start replication when the source and target data is in a synchronized state, where the corresponding rows in the source and target tables contain identical data

values. Unless you are starting with brand new source and target databases with no current user activity, you will need to activate change capture and apply processes to handle ongoing transactional changes while an initial load is being applied to the target. This process is known as *initial synchronization*, or also as *instantiation*. The initial load captures a point-in-time snapshot of the source data and applies it to the target, while Oracle GoldenGate maintains any changes that are made after that point.

See Instantiating Oracle GoldenGate with an Initial Load in *Administering Oracle GoldenGate* for instantiation options.

- Starting Extract During Instantiation
- Changing the Position of Extract to a Later Time

Starting Extract During Instantiation

When Extract starts for the first time to begin capturing data during the instantiation process, it captures all of the transaction data that it encounters after the specified start point, but none of the data that occurred before that point. To ensure that Extract does not start in the middle of ongoing transactions that would be discarded, set the tables that are to be captured to an inactive state. You can either put the system into a restricted state by using the `ALCOBJ` command to lock the objects or libraries, or you can force all of the current transactions on those tables to stop at a certain point.

After initialization is complete, remember to unlock any objects that you locked. To do so, log off of the session that locked the objects or use the `DLCOBJ` command from the OS/400 command line.

Changing the Position of Extract to a Later Time

You may at some point, over the life of an Extract run, need to set the position of Extract in the data stream manually. To reposition Extract, use the `ALTER EXTRACT` command in GGSCI. To help you identify any given Extract read position, the `INFO EXTRACT` command shows the positions for each journal in an Extract configuration, including the journal receiver information. See *Reference for Oracle GoldenGate* to know more.

Testing Your Configuration

It is important to test your configuration in a test environment before deploying it 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.

7

Instantiating and Starting Oracle GoldenGate Replication

This chapter contains instructions for configuring an initial load of target data, adding the required processes to instantiate replication, and perform the instantiation. The expected outcome of these steps is that source-target data is made consistent (known as the initial synchronization), and that Oracle GoldenGate captures and delivers ongoing transactional changes so that consistency is maintained going forward.

Topics:

About the Instantiation Process

During the initialization of the Oracle GoldenGate environment, you will be doing an initial data synchronization and starting the Oracle GoldenGate processes for the first time. In conjunction with those procedures, you will be creating the process groups for which you created parameter files in Configuring Oracle GoldenGate for DB2 for i .

To create an Extract process group, an initial start position for data capture must be established. This initial position will be based on a transaction boundary that is based on either of the following:

- a timestamp
- the end of the journal(s)
- A specific system sequence number
- A specific sequence number in the journal(s)

When Extract starts for the first time to begin capturing data, it captures all of the transaction data that it encounters after the specified start point, but none of the data that occurred before that point. To ensure that Extract does not start in the middle of ongoing transactions that would be discarded, set the tables that are to be captured to an inactive state. You can either put the system into a restricted state by using the `ALCOBJ` command to lock the objects

or libraries, or you can force all of the current transactions on those tables to stop at a certain point.

After initialization is complete, remember to unlock any objects that you locked. To do so, log off of the session that locked the objects or use the `DLCOBJ` command from the OS/400 command line.

Overview of Basic Oracle GoldenGate Instantiation Steps

These instructions show you how to instantiate the basic replication environment that you configured in Chapter 4. These steps are:

- Satisfying Prerequisites for Instantiation
- Making the Instantiation Procedure More Efficient
- Configuring the Initial Load
- Adding Change-Capture and Change-Delivery processes
- Performing the Target Instantiation
- Monitoring Processing after the Instantiation
- Backing up Your Oracle GoldenGate Environment
- Positioning Extract After Startup

Satisfying Prerequisites for Instantiation

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

- Configure Change Capture and Delivery
- Add Collision Handling
- Prepare the Target Tables

Configure Change Capture and Delivery

By the time you are ready to instantiate the replication environment, all of your Extract and Replicat process groups must be configured with completed parameter files as directed in "Configuring Oracle GoldenGate for DB2 for i ".

In addition, all of the other setup requirements in this manual must be satisfied.

Add Collision Handling

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

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

- `INSERT` operations for which the row already exists.

- `UPDATE` and `DELETE` operations for which the row does not exist.

For more information about this parameter, see the Oracle GoldenGate *Windows and UNIX Reference Guide*.

Prepare the Target Tables

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

- Data: Make certain that the target tables are empty. Otherwise, there may be duplicate-row errors or conflicts between existing rows and rows that are being loaded.
- Constraints: If you have not done so already, disable foreign-key constraints and check constraints. Foreign-key constraints can cause errors, and check constraints can slow down the loading process.
- Indexes: Remove indexes from the target tables. Indexes are not necessary for the inserts performed by the initial load process and will slow it down significantly. You can add back the indexes after the load is finished.
- Keys: To use the `HANDLECOLLISIONS` function to reconcile incremental data changes with the load, each target table must have a primary or unique key. If you cannot create a key through your application, use the `KEYCOLS` option of the `TABLE` and `MAP` parameters to specify columns as a substitute key for Oracle GoldenGate's purposes. If you cannot create keys, the affected source table must be quiesced for the load.

Making the Instantiation Procedure More Efficient

The following are some suggestions for making the instantiation process move more efficiently.

- Share Parameters Between Process Groups
- Use Parallel Processes

Share Parameters Between Process Groups

Some of the parameters that you use in a change-synchronization parameter file also are required in an initial-load Extract and initial-load Replicat parameter file. To take advantage of the commonalities, you can use any of the following methods:

- Copy common parameters from one parameter file to another.
- Store the common parameters in a central file and use the `OBEY` parameter in each parameter file to retrieve them.
- Create an Oracle GoldenGate macro for the common parameters and then call the macro from each parameter file with the `MACRO` parameter.

Use Parallel Processes

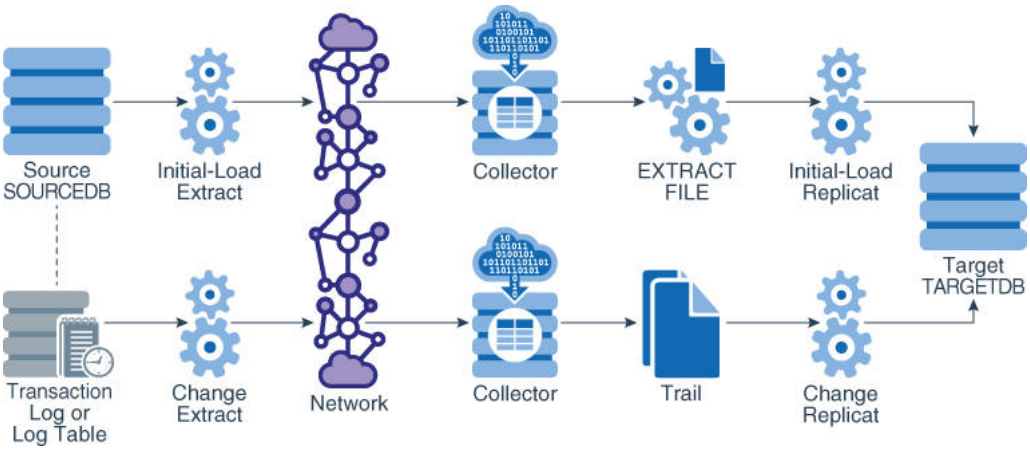
You can configure parallel initial-load processes to perform the initial load more quickly. It is important to keep tables with foreign-key relationships within the same set of processes. You can isolate large tables from smaller ones by using different sets of processes, or simply apportion the load across any number of process sets. To configure parallel processes correctly, see Administering Oracle GoldenGate for Windows and UNIX.

Configuring the Initial Load

Oracle GoldenGate supports the following load methods specifically for Oracle:

- Configuring an Initial Load from File to Replicat
- Configuring an initial load with a database utility

Configuring an Initial Load from File to Replicat



To use Replicat to establish the target data, you use an initial-load Extract to extract source records from the source tables and write them to an extract file in canonical format. From the file, an initial-load Replicat loads the data using the database interface. During the load, the change-synchronization groups extract and replicate incremental changes, which are then reconciled with the results of the load.

During the load, the records are applied to the target database one record at a time, so this method may be considerably slower than using a native DB2 for i load utility. This method permits data transformation to be done on either the source or target system.

To Configure a Load from File to Replicat

1. On the source and target systems, run GGSCI and start Manager.

   ```
   START MANAGER
   ```

2. On the source system, issue the following command to create an initial-load Extract parameter file. This Extract should have a different name from the Extract groups that capture the transactional data.

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

3. Enter the parameters listed in the following table in the order shown, starting a new line for each parameter statement.

Parameter	Description
SOURCEISTABLE	Designates Extract as an initial load process extracting records directly from the source tables.

Parameter	Description
SOURCEDB *database* USERID *user id*, PASSWORD *password*, BLOWFISH ENCRYPTKEY *keyname*	Specifies database connection information. • SOURCEDB specifies the name of the source database. • USERID specifies the Extract database user profile. • PASSWORD specifies the user's password that was encrypted with the ENCRYPT PASSWORD command (see "Encrypting the Extract and Replicat Passwords"). Enter or paste the encrypted password after the PASSWORD keyword. • BLOWFISH ENCRYPTKEY *keyname* specifies the name of the lookup key in the local ENCKEYS file.
RMTHOST *hostname*, MGRPORT *portnumber*, [*encryption options*]	• RMTHOST specifies the name or IP address of the target system. • MGRPORT specifies the port number where Manager is running on the target. • *encryption options* specifies optional encryption of data across TCP/IP. For additional options and encryption details, see Reference for Oracle GoldenGate for Windows and UNIX.
ENCRYPTTRAIL BLOWFISH KEYNAME *keyname*	Encrypts the remote file with Blowfish encryption. For more information about security, see Administering Oracle GoldenGate for Windows and UNIX.
RMTFILE *path name*, [MEGABYTES *n*] • *path name* is the relative or fully qualified name of the file. • MEGABYTES designates the size of each file.	Specifies the remote file to which the load data will be written. Oracle GoldenGate creates this file during the load. **Note**: The size of an extract file cannot exceed 2GB.
TABLE *owner.table*; • owner is the library or schema name. • *table* is the name of the table or a group of tables defined with wildcards. To exclude tables from a wildcard specification, use the TABLEEXCLUDE parameter.	Specifies a source table or tables for initial data extraction.

4. Enter any appropriate optional Extract parameters listed in Reference for Oracle GoldenGate for Windows and UNIX.
5. Save and close the parameter file.
6. On the target system, issue the following command to create an initial-load Replicat parameter file. This Replicat should have a different name from the Replicat group that applies the transactional data.

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

7. Enter the parameters listed in Table 7-1 in the order shown, starting a new line for each parameter statement.

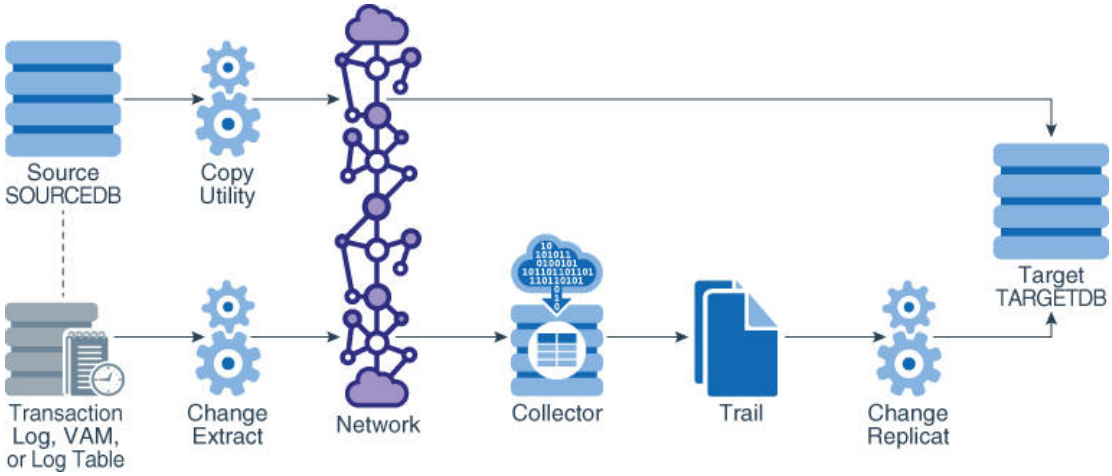
Table 7-1 Initial Load Replicat Parameters for Loading Data from File to Replicat

Parameter	Description
`SPECIALRUN`	Implements the initial-load Replicat as a one-time run that does not use checkpoints.
`END RUNTIME`	Directs the initial-load Replicat to terminate when the load is finished.
`TARGETDB` *`database`*`, USERID` *`user id`*`, PASSWORD` *`pw`*`,` *`algorithm`* `ENCRYPTKEY` *`keyname`*	Specifies database connection information. • TARGETDB specifies the Data Source Name that is defined for the DB2 for i target database through the ODBC interface on the Windows or Linux system. • `USERID` specifies the Replicat database user profile. • `PASSWORD` specifies the user's password that was encrypted with the `ENCRYPT PASSWORD` command (see "Encrypting the Extract and Replicat Passwords"). Enter or paste the encrypted password after the `PASSWORD` keyword. • *`algorithm`* `ENCRYPTKEY` *`keyname`* specifies the encryption method and keyname that was specified in the `ENCRYPT PASSWORD` command.
`DECRYPTTRAIL BLOWFISH KEYNAME` *`keyname`*	Decrypts the input trail. `BLOWFISH` is required because this is the algorithm that is supported to encrypt the file from DB2 for i.
`EXTFILE` *`path name`* `|` `EXTTRAIL` *`path name`* • *`path name`* is the relative or fully qualified name of the file or trail. • Use `EXTTRAIL` only if you used the `MAXFILES` option of the `RMTFILE` parameter in the Extract parameter file.	Specifies the input extract file specified with the Extract parameter `RMTFILE`.
`SOURCEDEFS` *`file name`* `|` `ASSUMETARGETDEFS` • Use `SOURCEDEFS` if the source and target tables have different definitions. Specify the relative or fully qualified name of the source-definitions file generated by the `DEFGEN` utility. • Use `ASSUMETARGETDEFS` if the source and target tables have the same definitions.	Specifies how to interpret data definitions.
`MAP` *`owner.table`*`,` `TARGET` *`owner.table`*`;` • *`owner`* is the schema name. • *`table`* is the name of a table or a wildcard definition for multiple tables. To exclude tables from a wildcard specification, use the *`MAPEXCLUDE`* parameter.	Specifies a relationship between a source and target table or tables.

8. Enter any appropriate optional Replicat parameters listed in the Reference for Oracle GoldenGate for Windows and UNIX.

9. Save and close the file.

Configuring an initial load with a database utility



This graphic shows the parallel flows of the initial load and the ongoing capture and replication of transactional changes during the load period. The copy utility writes the data to a file, which is loaded to the target. Meanwhile, an Extract process captures change data and sends it to a trail on the target for Replicat to read and apply to the target.

For an initial load between two DB2 for i source and target systems, you can use the DB2 for i system utilities to establish the target data. To do this, you save the file(s) that you want to load to the target by using the `SAVOBJ` or `SAVLIB` commands, and then you restore them on the target using the `RSTOBJ` or `RSTLIB` commands.

Another alternative is to use the DB2 for i commands `CPYTOIMPF` (Copy to Import File) and `CPYFRMIMPF` (Copy from Import File) to create files that can be used with the bulk load utilities of other databases. See the DB2 for i Information Center documentation for more details on "Copying between different systems."

In both cases, no special configuration of any Oracle GoldenGate initial-load processes is needed. You use the change-synchronization process groups that you configured in Configuring Oracle GoldenGate for DB2 for i . You start a change-synchronization Extract group to extract ongoing data changes while you are making the copy and loading it. When the copy is finished, you start the change-synchronization Replicat group to re-synchronize rows that were changed while the copy was being applied. From that point forward, both Extract and Replicat continue running to maintain data synchronization. See "Adding Change-Capture and Change-Delivery processes".

Adding Change-Capture and Change-Delivery processes

> **Note:**
>
> Perform these steps at or close to the time that you are ready to start the initial load and change capture.

These steps establish the Oracle GoldenGate Extract, data pump, and Replicat processes that you configured in Configuring Oracle GoldenGate for DB2 for i . Collectively known as the "change-synchronization" processes, these are the processes that:

- capture and apply ongoing source changes while the load is being performed on the target
- reconcile any collisions that occur

> **Note:**
>
> Perform these steps as close as possible to the time that you plan to start the initial load processes. You will start these processes during the initial load steps.

- Add the Primary Extract
- Add the Local Trail
- Add the Data Pump Extract Group
- Add the Remote Trail
- Add the Replicat Group

Add the Primary Extract

These steps add the primary Extract that captures change data.

- Understanding the Primary Extract Start Point
- Establishing the Required and Optional Extract Start Points

Understanding the Primary Extract Start Point

When you add the primary Extract group, you establish an initial start position for data capture. This initial position can be a transaction boundary that is based on either of the following:

- a timestamp
- the end of the journal(s)
- a specific *system* sequence number
- a specific *journal* sequence number (per journal)

The options that are available to you assume a global start point and optional journal-specific start points.

- To position by a timestamp, at the end of the journals, or at a system sequence number, you will use the `ADD EXTRACT` command with the appropriate option. This command establishes a global start point for all journals and is a required first step.

- After issuing the `ADD EXTRACT` command, you can then optionally position any *specific* journal at a specific journal sequence number by using the `ALTER EXTRACT` command with an appropriate journal option.

Establishing the Required and Optional Extract Start Points

These steps include the `ADD EXTRACT` and `ALTER EXTRACT` commands to enable you to establish your desired start points.

1. Run GGSCI.
2. Issue the `ADD EXTRACT` command to add the primary Extract group and establish the global start point.

```
ADD EXTRACT group name, TRANLOG
{
, BEGIN {NOW | yyyy-mm-dd[hh:mi:[ss[.cccccc]]]} |
, EOF |
, SEQNO seqno
}
```

   Where:

   - *group name* is the name of the primary Extract group that captures the transactional changes.
   - `TRANLOG` specifies the journals as the data source.
   - `BEGIN` specifies to begin capturing data as of a specific *time*. Select one of two options: `NOW` starts at the first record that is timestamped at the same time that `BEGIN` is issued. *yyyy-mm-dd[hh:mi:[ss[.cccccc]]]* starts at an explicit timestamp. Logs from this timestamp must be available.
   - `SEQNO seqno` specifies to begin capturing data at, or just after, a system sequence number, which is a decimal number up to 20 digits in length.

3. (Optional) Issue the following command to alter any `ADD EXTRACT` start position to set the start position for a specific journal in the same Extract configuration. A *specific* journal position set with `ALTER EXTRACT` does not affect any global position that was previously set with `ADD EXTRACT` or `ALTER EXTRACT`; however a *global* position set with `ALTER EXTRACT` overrides any specific journal positions that were previously set in the same Extract configuration.

```
ALTER EXTRACT group name,
{
ALTER EXTRACT {BEGIN {NOW | yyyy-mm-dd [hh:mi:[ss[.cccccc]]] [JOURNAL
journal_library/journal_name [[JRNRCV receiver_library/receiver_name]] |
, EOF [JOURNAL journal_library/journal_name [[JRNRCV receiver_library/
receiver_name]] |
, SEQNO seqno [JOURNAL journal_library/journal_name [[JRNRCV receiver_library/
receiver_name]]
}
```

> **Note:**
>
> SEQNO, when used with a journal in ALTER EXTRACT, is the journal sequence number that is relative to that specific journal, not the system sequence number that is global across journals.

Example 7-1 Timestamp Start Point

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

Example 7-2 NOW Start Point

```
ADD EXTRACT finance, TRANLOG, BEGIN NOW
```

Example 7-3 System Sequence Number Start Point

```
ADD EXTRACT finance, TRANLOG, SEQNO 2952
```

Example 7-4 Journal Start Point

```
ALTER EXTRACT finance, SEQNO 1234  JOURNAL accts/acctsjrn
```

Example 7-5 Journal and Receiver Start Point

```
ALTER EXTRACT finance, SEQNO 1234 JOURNAL accts/acctsjrn JRNRCV accts/jrnrcv0005
```

Add the Local Trail

This step adds 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.

Example 7-6

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

Add the Data Pump Extract Group

This step adds 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 data-pump Extract group.
- EXTTRAILSOURCE *trail name* is the relative or fully qualified name of the local trail.

Example 7-7

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

Add the Remote Trail

This step adds 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.

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, and *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.

Example 7-8

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

Add the Replicat Group

These steps add the Replicat group that reads the remote trail (which gets created automatically on the target) and applies the data changes to the target Oracle database.

1. Run GGSCI on the target system.
2. Issue the ADD REPLICAT command.

   ```
   ADD REPLICAT group name, EXTTRAIL pathname
   ```

   Where:

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

Example 7-9

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

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.

- To Perform Instantiation from File to Replicat
- To Perform Instantiation with a Database Utility

To Perform Instantiation from File to Replicat

1. Make certain that you have addressed the requirements in Satisfying Prerequisites for Instantiation.
2. On the source and target systems, run GGSCI and start the Manager process.

   ```
   START MANAGER
   ```

3. On the source system, start the primary and data pump Extract groups to start change extraction.

   ```
   START EXTRACT primary Extract group name
   START EXTRACT data pump Extract group name
   ```

4. From the directory where Oracle GoldenGate is installed on the source system, start the initial-load Extract as follows:

   ```
   $ /GGS directory/extract paramfile dirprm/initial-load Extract name.prm
   reportfile path name
   ```

   Where: *initial-load Extract name* is the name of the initial-load Extract that you used when creating the parameter file, and *path name* is the relative or fully qualified name of the Extract report file (by default the *dirrpt* sub-directory of the Oracle GoldenGate installation directory).
5. Verify the progress and results of the initial extraction by viewing the Extract report file using the operating system's standard method for viewing files.
6. Wait until the initial extraction is finished.
7. On the target system, start the initial-load Replicat.

   ```
   $ /GGS directory/replicat paramfile dirprm/initial-load Replicat
   name.prm reportfile path name
   ```

   Where: *initial-load Replicat name* is the name of the initial-load Replicat that you used when creating the parameter file, and *path name* is the relative or fully qualified name of the Replicat report file (by default the *dirrpt* sub-directory of the Oracle GoldenGate installation directory).
8. When the initial-load Replicat is finished running, verify the results by viewing the Replicat report file using the operating system's standard method for viewing files.
9. On the target system, start change replication.

   ```
   START REPLICAT Replicat group name
   ```

10. On the target system, issue the following command to verify the status of change replication.

    ```
    INFO REPLICAT Replicat group name
    ```

11. Continue to issue the `INFO REPLICAT` command until you have verified 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 point.
12. On the target system, issue the following command to turn off the `HANDLECOLLISIONS` parameter and disable the initial-load error handling.

    ```
    SEND REPLICAT Replicat group name, NOHANDLECOLLISIONS
    ```

13. On the target system, edit the Replicat parameter file to remove the `HANDLECOLLISIONS` parameter. This prevents `HANDLECOLLISIONS` from being enabled again the next time Replicat starts.

> **Caution:**
>
> Do not use the `VIEW PARAMS` or `EDIT PARAMS` command to view or edit an existing parameter file that is in a character set other than that of the local operating system (such as one where the `CHARSET` option was used to specify a different character set). View the parameter file from outside GGSCI if this is the case; otherwise, the contents may become corrupted.

14. Save and close the parameter file.

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

To Perform Instantiation with a Database Utility

1. Make certain that you have addressed the requirements in "Satisfying Prerequisites for Instantiation".
2. On the source and target systems, run GGSCI and start the Manager process.

   ```
   START MANAGER
   ```

3. On the source system, start the primary and data pump Extract groups to start change extraction.

   ```
   START EXTRACT primary Extract group name
   START EXTRACT data pump Extract group name
   ```

4. On the source system, start making the copy.
5. Wait until the copy is finished and record the time of completion.
6. View the Replicat parameter file to make certain that the `HANDLECOLLISIONS` parameter is listed. If not, edit the file and add the parameter to the file.

   ```
   EDIT PARAMS Replicat group name
   ```

> **Note:**
>
> Do not use the `VIEW PARAMS` or `EDIT PARAMS` command to view or edit an existing parameter file that is in a character set other than that of the local operating system (such as one where the `CHARSET` option was used to specify a different character set). View the parameter file from outside GGSCI if this is the case; otherwise, the contents may become corrupted.

7. On the target system, start change replication.

   ```
   START REPLICAT Replicat group name
   ```

8. On the target system, issue the following command to verify the status of change replication.

   ```
   INFO REPLICAT Replicat group name
   ```

9. Continue to issue the `INFO REPLICAT` command until you have verified that change replication has posted all of the change data that was generated during the initial load. Reference the time of completion that you recorded. For example, if the copy stopped at 12:05, make sure change replication has posted data up to that point.
10. On the target system, issue the following command to turn off the `HANDLECOLLISIONS` parameter and disable the initial-load error handling.

    ```
    SEND REPLICAT Replicat group name, NOHANDLECOLLISIONS
    ```

11. On the target system, edit the Replicat parameter file to remove the `HANDLECOLLISIONS` parameter. This prevents `HANDLECOLLISIONS` from being enabled again the next time Replicat starts.

> **Caution:**
>
> Do not use the `VIEW PARAMS` or `EDIT PARAMS` command to view or edit an existing parameter file that is in a character set other than that of the local operating system (such as one where the `CHARSET` option was used to specify a different character set). View the parameter file from outside GGSCI if this is the case; otherwise, the contents may become corrupted.

12. Save and close the parameter file.

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

Monitoring Processing after the Instantiation

After the target is instantiated and replication is in effect, you should view the status, lag, and overall health of the replication environment to ensure that processes are running properly, that there are no warnings in the Oracle GoldenGate error log, and that lag is at an acceptable level. You can view Oracle GoldenGate processes from:

- GGSCI: For information about monitoring processes, see Administering Oracle GoldenGate for Windows and UNIX.
- Oracle GoldenGate Monitor: See the administration documentation and online help for that product. Oracle GoldenGate Monitor provides a graphical-based monitoring environment for all of your Oracle GoldenGate instances.

You also should verify that capture and delivery is being performed for all of the tables in the Oracle GoldenGate configuration, and that the source and target data are synchronized. You can use the Oracle GoldenGate Veridata product for this purpose.

Backing up Your 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. This directory contains critical sub-directories, files and programs.

The most critical working files in this directory consume the vast majority of backup space; therefore it makes sense just to back up the entire installation directory for fast, simple recovery.

Positioning Extract After Startup

You may at some point, over the life of an Extract run, need to set the position of Extract in the data stream manually. To reposition Extract, use the `ALTER EXTRACT` command in GGSCI. To help you identify any given Extract read position, the `INFO EXTRACT` command shows the positions for each journal in an Extract configuration, including the journal receiver information. For more information about these commands, see Reference for Oracle GoldenGate for Windows and UNIX.

> **Note:**
>
> Because the extract will be synchronizing all of the journals in the extract by system sequence number because it is possible for a transaction to be split across them, if a given journal is independently repositioned far into the past, the resulting latency from reprocessing the entries will cause the already-read journals to stall until the reading of the latent journal catches up.

8
Using Remote Journal

This chapter contains instructions for remote journal preparation and adding a remote journal. Remote Journal support in the IBM DB2 for i operating system provides the ability for a system to replicate, in its entirety, a sequence of journal entries from one DB2 for i system to another. Once setup, this replication is handled automatically and transparently by the operating system. The entries that are replicated are placed in a journal on the target system that is available to be read by an application in the same way as on the source system. You must have an understanding of how to setup and use remote journaling on an DB2 for i system to use this feature with Oracle GoldenGate. There are no special software requirements for either Oracle GoldenGate or the DB2 for i systems to use remote journaling.

Topics:

Preparing to Use Remote Journals

Before establishing the remote journal environment, complete the following steps:

1. Determine the extent of your remote journal network or environment.
2. *Library redirection* is the ability to allow the remote journal and associated journal receivers to reside in different libraries on the target system from the corresponding source journal and its associated journal receivers.

   Determine what library redirection, if any, you will be using for the remote journals and associated journal receivers.
3. Ensure that all selected libraries exist on the target systems. You must consider whether or not library redirection will be used when adding the remote journal.
4. Create the appropriate local journal if it does not already exist.
5. Configure and activate the communications protocol you have chosen to use.
6. After you have configured the communications protocol, it must be active while you are using the remote journal function.

   For example, if you are using the OptiConnect for IBM i bus transport method, then the OptiConnect for IBM i subsystem, QSOC, must be active. QSOC must be active for both the source system and the target system, and the appropriate controllers and devices must be varied on. If you are using a SNA communications transport, vary on the appropriate line, controller, and devices and ensure subsystem QCMN is active on both systems. Start of change If you are using TCP/IP or Sockets IPv6, you must start TCP/IP by using the Start TCP/IP (`STRTCP`) command, including the distributed data management (DDM) servers. If you are using data port, you must configure a cluster, make sure that the cluster is active, and start the internet Daemon (`inetd`) server using the Start TCP/IP Server (`STRTCPSVR`) command.End of change
7. If one does not already exist, create the appropriate relational database (RDB) directory entry that will be used to define the communications protocol for the remote journal

environment. When TCP communications are being used to connect to an independent disk pool, the RDB entry to the independent disk pool must have the Relational database value set to the target system's local RDB entry and the relational database alias value set to the independent disk pool's name.

8. Now you should be able to see the remote database connection by issuing the `WRKRDBDIRE` command.

```
 Work with Relational Database Directory Entries

 Position to . . . . . .

 Type options, press Enter.
   1=Add 2=Change 4=Remove 5=Display details 6=Print details

                               Remote
 Option Entry Location Text

         SYS1 system1
         SYS2 system2
         MYSYSTEM *LOCAL Entry added by system



 Bottom
  F3=Exit F5=Refresh F6=Print list F12=Cancel F22=Display entire field
  (C) COPYRIGHT IBM CORP. 1980, 2007.
```

Adding a Remote Journal

Adding a remote journal creates a remote journal on a target system or independent disk pool and associates that remote journal with the journal on the source system. This occurs if this is the first time the remote journal is being established for a journal. The journal on the source system can be either a local or remote journal.

If a remote journal environment has previously been established, adding a remote journal reassociates the remote journal on the target system with the journal on the source system.

You can establish and associate a remote journal on a target system with a journal on the source system by one of the following methods:

- System i Navigator.
- Add the Remote Journal (`QjoAddRemoteJournal`) API on the source system.
- Add the Remote Journal (`ADDRMTJRN`) command on the source system.
- What Happens During Add Remote Journal Processing?
- Guidelines For Adding a Remote Journal

What Happens During Add Remote Journal Processing?

The processing that takes place as part of adding a remote journal includes the following:

- A check is performed on the target system to verify that the user profile adding the remote journal exists. A user profile with the same name as the user profile which is adding a remote journal must exist on the target system. If the profile does not exist on the target system, then an exception is signaled, and the processing ends.
- A check is performed to verify that the target system has a library by the same name as the library for the journal on the source system. If the library does not exist on the target system, then an exception is signaled, and the processing ends.
- A check is performed on the target system to determine if a journal by the same qualified name as the journal on the source system already exists. If a journal already exists, it can be used for the remainder of the add remote journal processing if it meets the following criteria:
  1. It is a remote journal.
  2. It was previously associated with this same source journal or part of the same remote journal network.
  3. The type of the remote journal matches the specified remote journal type.
- If a journal was found, but does not meet the preceding criteria, then an exception is signaled, and the processing ends. Otherwise, the remote journal is used for the rest of the add remote journal processing.
- If no journal is found on the specified target system, then a remote journal is created on the target system. The new remote journal has the same configuration, authority, and audit characteristics of the source journal. The journal that is created has a journal type of `*REMOTE`.

When adding the remote journal, you must specify the type of remote journal to add. The remote journal type influences the library redirection rules and other operational characteristics for the journal.

Guidelines For Adding a Remote Journal

You should observe the following guidelines for adding a remote journal:

- You can only associate a remote journal with a single source journal.

  Note: The same remote journal can then have additional remote journals that are associated with it that are located on other target systems. This is the cascade configuration that is shown in Network configurations for remote journals.
- The remote journal will only have its attached receiver populated with journal entries that are replicated from the corresponding journal receiver on the source system. No journal entries can be directly deposited to a remote journal.
- A maximum of 255 remote journals can be associated with a single journal on a source system. This can be any combination of asynchronously maintained or synchronously maintained remote journals.

To Add a Remote Journal

The following is an example using the physical file QGPL/TESTPF setup to have remote journaling enabled to a second system.

1. Create the physical file.:

```
> CRTPF FILE(QGPL/TESTPF) RCDLEN(10)
  File TESTPF created in library QGPL.
  Member TESTPF added to file TESTPF in QGPL.
```

2. Create the local journal receiver and journals, and enable the journaling of the physical file created:

```
> crtjrnrcv jrnrcv(qgpl/jrcvrmt)
  Journal receiver JRCVRMT created in library QGPL

> crtjrn jrn(qgpl/jrnrmt) jrnrcv(qgpl/jrcvrmt) fixlendta(*job *usr *pgm
*sysseq)
  Journal JRNRMT created in library QGPL

strjrnpf file(qgpl/testpf) jrn(qgpl/testpf)
  1 of 1 files have started journaling
```

3. Add the remote journal:

```
> addrmtjrn rdb(sys2) srcjrn(qgpl/JRNRMT) rmtjrntype(*TYPE2)
  Remote journal JRNRMT in QGPL was added
```

4. Activate the remote journaling:

```
> chgrmtjrn rbd(sys2) srcjrn(qgpl/jrnrmt) jrnstate(*active)
  Remote journal JRNRMT in library QGPL was activated
```

Part III
Using Oracle GoldenGate for DB2 for z/OS

Oracle GoldenGate for DB2 for z/OS supports capture and delivery of initial load and transactional data for supported DB2 for z/OS database versions.

Oracle GoldenGate for DB for z/OS supports the mapping, filtering, and transformation of source data, unless noted otherwise in this document, along with replicating data derived from other source databases supported by Oracle GoldenGate, into DB2 for z/OS databases.

Oracle GoldenGate for DB2 for z/OS is installed and runs remotely on Linux and zLinux.

Topics:

9

Understanding What's Supported for DB2 for z/OS

This chapter contains information on database and table features supported byOracle GoldenGate for DB2 z/OS.

Topics:

Supported DB2 for z/OS Data Types

This section lists the DB2 for z/OS data types that Oracle GoldenGate supports and any limitations of this support.

- Oracle GoldenGate does not perform character set conversion for columns that could contain multi-byte data. This includes `GRAPHIC`, `VARGRAPHIC` and `DBCLOB` data types, as well as `CHAR`, `VARCHAR`, and `CLOB` for tables defined with `ENCODING_SCHEME` of '`M`' (multiple CCSID set or multiple encoding schemes) or '`U`' (Unicode). Such data is only supported if the source and target systems are the same CCSID.
- Oracle GoldenGate supports ASCII, EBCDIC, and Unicode data format. Oracle GoldenGate converts between ASCII and EBCDIC data automatically. Unicode is not converted.
- Oracle GoldenGate supports most DB2 data types except those listed in Non-Supported DB2 for z/OS Data Types.

Limitations of Support

- The support of range and precision for floating-point numbers depends on the host machine. In general, the precision is accurate to 16 significant digits, but you should review the database documentation to determine the expected approximations. Oracle GoldenGate rounds or truncates values that exceed the supported precision.
- Oracle GoldenGate does not support the filtering, column mapping, or manipulation of large objects greater than 4K in size. You can use the full Oracle GoldenGate functionality for objects that are 4K or smaller.
- Oracle GoldenGate supports the default `TIMESTAMP` and the `TIMESTAMP` with `TIMEZONE` to up to 9 digit fractional value, but no further.

Non-Supported DB2 for z/OS Data Types

This section lists DB2 for z/OS data types that Oracle GoldenGate does not support. Data that is not supported may affect the integrity of the target data in relation to the source data.

- XML
- User-defined types
- Negative dates

Supported Objects and Operations for DB2 z/OS

This section lists the database objects and types of operations that Oracle GoldenGate supports.

- Parallel Replicat is supported with Oracle GoldenGate for DB2 z/OS.
- Extraction and replication of DML operations on DB2 for z/OS tables that contain rows of up to 512KB in length. This size exceeds the maximum row size of DB2.
- INSERT operations from the IBM LOAD utility are supported for change capture if the utility is run with LOG YES and SHRLEVEL CHANGE, and the source tables that are being loaded have DATA CAPTURE CHANGES enabled (required by Oracle GoldenGate) and are specified in the Oracle GoldenGate Extract configuration. Oracle GoldenGate also supports initial loads with the LOAD utility to instantiate target tables during initial synchronization.
- Oracle GoldenGate supports the maximum number of columns per table, which is supported by the database.
- Oracle GoldenGate supports the maximum column size that is supported by the database.
- Extraction and replication of data that is stored using DB2 data compression (CREATE TABLESPACE COMPRESS YES).
- Capture from temporal history tables is supported.
- TRUNCATE TABLE is supported, but because this command issues row deletes to perform the truncate, they are shown in Oracle GoldenGate statistics as such, and not as a truncate operation. To replicate a TRUNCATE , the Replicat process uses a DELETE operation without a WHERE clause.
- TRUNCATES are always captured from a DB2 for z/OS source, but can be ignored by Replicat if the IGNORETRUNCATES parameter is used in the Replicat parameter file.
- UNICODE columns in EBCDIC tables are supported.
- Supported options with SHOWTRANS

  ```
  SHOWTRANS [transaction_ID]  [COUNT n]
  [DURATION duration unit]
  [FILE file_name] |
  ```

  transaction_ID and count cannot be specified together.

  transaction_ID and duration cannot be specified together.

- Options supported with SKIPTRANS and FORCETRANS:

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

Non-Supported Objects and Operations for DB2 for z/OS

The following objects and operations are not supported by Oracle GoldenGate on DB2 for z/OS:

- Extraction or replication of DDL operations
- Clone tables
- Data manipulation, including compression, that is performed within user-supplied DB2 exit routines, such as:
  - Date and time routines
  - Edit routines (`CREATE TABLE EDITPROC`)
  - Validation routines (`CREATE TABLE VALIDPROC`)
- Replicating with `BATCHSQL` is not fully functional for DB2 for z/OS. Non-insert operations are not supported so any update or delete operations will cause Replicat to drop temporarily out of `BATCHSQL` mode. The transactions will stop and errors will occur.

10

Preparing the DB2 for z/OS Database for Oracle GoldenGate

Learn how to prepare your database and environment to support Oracle GoldenGate.
Topics:

Preparing Tables for Processing

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

Disabling Triggers and Cascade Constraints

Disable triggers, cascade delete constraints, and cascade update constraints on the target tables, or alter them to ignore changes made by the Oracle GoldenGate database user. Oracle GoldenGate replicates DML that results from a trigger or cascade constraint. If the same trigger or constraint gets activated on the target table, it becomes redundant because of the replicated version, and the database returns an error. Consider the following example, where the source tables are `emp_src` and `salary_src` and the target tables are `emp_targ` and `salary_targ`.

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

Ensuring Row Uniqueness for Tables

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

Unless a KEYCOLS clause is used in the TABLE or MAP statement, Oracle GoldenGate selects a row identifier to use in the following order of priority:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. If none of the preceding key types exist (even though there might be other types of keys defined on the table) Oracle GoldenGate constructs a pseudo key of all columns that the database allows to be used in a unique key, excluding those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration.

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

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

- Using KEYCOLS to Specify a Custom Key

Using KEYCOLS to Specify a Custom Key

If a table does not have one of the preceding types of row identifiers, or if you prefer those identifiers not to be used, you can define a substitute key if the table has columns that always contain unique values. You define this substitute key by including a KEYCOLS clause within the Extract TABLE parameter and the Replicat MAP parameter. The specified key will override any existing primary or unique key that Oracle GoldenGate finds. For more information, see *Reference for Oracle GoldenGate*.

Handling ROWID Columns

Any attempt to insert into a target table that includes a column with a data type of ROWID GENERATED ALWAYS (the default) will fail with the following ODBC error:

```
ODBC error: SQLSTATE 428C9 native database error -798. {DB2 FOR OS/390}
{ODBC DRIVER}{DSN08015} DSNT408I SQLCODE = -798, ERROR: YOU CANNOT
INSERT A VALUE INTO A COLUMN THAT IS DEFINED WITH THE OPTION GENERATED
ALWAYS. COLUMN NAME ROWIDCOL.
```

You can do one of the following to prepare tables with ROWID columns to be processed by Oracle GoldenGate:

- Ensure that any `ROWID` columns in target tables are defined as `GENERATED BY DEFAULT`.
- If it is not possible to change the table definition, you can work around it with the following procedure.

To Work Around `ROWID GENERATE ALWAYS`:

1. For the source table, create an Extract `TABLE` statement, and use a `COLSEXCEPT` clause in that statement that excludes the `ROWID` column. For example:

   ```
   TABLE tab1, COLSEXCEPT (rowidcol);
   ```

   The `COLSEXCEPT` clause excludes the `ROWID` column from being captured and replicated to the target table.
2. For the target table, ensure that Replicat does not attempt to use the `ROWID` column as the key. This can be done in one of the following ways:
   - Specify a primary key in the target table definition.
   - If a key cannot be created, create a Replicat `MAP` parameter for the table, and use a `KEYCOLS` clause in that statement that contains any unique columns except for the `ROWID` column. Replicat will use those columns as a key. For example:

     ```
     MAP tab1, TARGET tab1, KEYCOLS (num, ckey);
     ```

Configure a Database Connection

This section contains instructions for setting up the Extract and Replicat connections to a DB2 z/OS database.

Topics:

Database Configuration for DB2 z/OS

No special DB2 z/OS database settings are required for Oracle GoldenGate.

Database User for Oracle GoldenGate Processes

Oracle GoldenGate requires a database user account. Create this account and assign privileges according to the following guidelines.

Assign the DB2 privileges listed in Table 10-1 to the user by which Extract and Replicat will be running. These are in addition to any permissions that DB2 ODBC requires. All Extract privileges apply to initial-load and log-based Extract processes, except where noted. The following authorities can be provided by granting either `SYSCTRL` or `DBADM` plus `SQLADM` authority to the user running the Oracle GoldenGate processes.

Table 10-1 Privileges Needed by Oracle GoldenGate for DB2 z/OS

User privilege	Extract	Replicat
MONITOR2 (does not apply to initial-load Extract)	X	
SELECT ON the following SYSIBM tables: SYSTABLES SYSCOLUMNS SYSTABLEPART SYSKEYS SYSINDEXES SYSCOLAUTH SYSDATABASE SYSFOREIGNKEYS SYSPARMS SYSRELS SYSROUTINES SYSSYNONYMS SYSTABAUTH SYSAUXRELS	X	X
SELECT on source tables[1]	X	
INSERT, UPDATE, DELETE on target tables		X
CREATE TABLE[2]		X
EXECUTE on ODBC plan (default is DSNACLI)	X	
Privileges required by SQLEXEC procedures or queries that you will be using.[3]	X	X

1 SELECT on source tables required only if tables contain LOB columns, or for an initial-load Extract, if used.
2 Required if using ADD CHECKPOINTTABLE in GGSCI to use the database checkpoint feature.
3 SQLEXEC enables stored procedures and queries to be executed by an Oracle GoldenGate process.

Setting Initialization Parameters

The following DB2 for z/OS initialization parameters apply to Oracle GoldenGate and must be set correctly before starting Oracle GoldenGate processes.

- MVSDEFAULTSSID: set to the DB2 subsystem.
- LOCATION: set to the DB2 location name as stored in the DB2 Boot Strap Dataset.
- MVSATTACHTYPE: set to RRSAF (Recoverable Resource Manager Services Attachment Facility) or CAF (Call Attachment Facility). IBM recommends using RRSAF.
- MULTICONTEXT: set to 1 if using RRSAF.
- PLANNAME: set to the DB2 plan. The default plan name is DSNACLI.

Do not use the `CURRENTAPPENSCH` initialization parameter (keyword).

> **Note:**
>
> When using the `CAF` attachment type, you must use the Oracle GoldenGate `DBOPTIONS` parameter with the `NOCATALOGCONNECT` option in the parameter file of any Extract or Replicat process that connects to DB2. This parameter disables the usual attempt by Oracle GoldenGate to obtain a second thread for the DB2 catalog. Otherwise, you will receive error messages, such as: `ODBC operation failed: Couldn't connect to` *`data source`* `for catalog queries.`

Specifying the Path to the Initialization File

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

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

Ensuring ODBC Connection Compatibility

To ensure that you configure the DB2 ODBC initialization file correctly, follow the guidelines in the *DB2 UDB for z/OS ODBC Guide and Reference* manual. One important consideration is the coding of the open and close square brackets (the [character and the] character). The square bracket characters are "variant" characters that are encoded differently in different coded character set identifiers (CCSID), but must be of the IBM-1047 CCSID in the ODBC initialization file. DB2 ODBC does not recognize brackets of any other CCSID. Note the following:

- The first (or open) bracket must use the hexadecimal characters `X'AD'` (`0xAD`).
- The second (or close) bracket must use the hexadecimal characters `X'BD'` (`0xBD`).

To set the correct code for square brackets, use any of the following methods.

- Use the `hex` command in OEDIT and change the hex code for each character appropriately.
- Use the `iconv` utility to convert the ODBC initialization file. For example, to convert from CCSID IBM-037 to IBM-1047, use the following command:

  ```
  iconv -f IBM-037 -t IBM-1047 ODBC.ini > ODBC-1047.ini

  mv ODBC-1047.ini ODBC.ini
  ```

- Change your terminal emulator or terminal configuration to use CCSID IBM-1047 when you create or alter the file.

Specifying the Number of Connection Threads

Every Oracle GoldenGate process makes a database connection. Depending on the number of processes that you will be using and the number of other DB2 connections that you expect, you might need to adjust the following DB2 system parameters on the DSNTIPE DB2 Thread Management Panel:

- `MAX USERS` (macro `DSN6SYSP CTHREAD`)
- `MAX TSO CONNECT` (macro `DSN6SYSP IDFORE`)
- `MAX BATCH CONNECT` (macro `DSN6SYSP IDBACK`)

If using RRSAF, allow:

- Two DB2 threads per process for each of the following:
  - Extract
  - Replicat
  - The GGSCI command `DBLOGIN` (logs into the database)
  - `DEFGEN` utility (generates data definitions for column mapping)
- One extra DB2 thread for Extract for IFI calls.
- One extra DB2 thread for each `SQLEXEC` parameter statement that will be issued by each Extract and Replicat process. For more information about `SQLEXEC`, see the *Reference for Oracle GoldenGate*.

If using CAF, there can be only one thread per Oracle GoldenGate process.

Monitoring Processes

These sections provide information about monitoring Oracle GoldenGate with z/OS system facilities.

- Interpreting Statistics for Update Operations

Interpreting Statistics for Update Operations

The actual number of DML operations that are executed on the DB2 database might not match the number of extracted DML operations that are reported by Oracle GoldenGate. DB2 does not log update statements if they do not physically change a row, so Oracle GoldenGate cannot detect them or include them in statistics.

Supporting Globalization Functions

Oracle GoldenGate provides globalization support and you should take into consideration when using this support.

- Replicating From a Source that Contains Both ASCII and EBCDIC
- Specifying Multi-Byte Characters in Object Names

Replicating From a Source that Contains Both ASCII and EBCDIC

When replicating to or from a DB2 source system to a target that has a different character set, some consideration must be given to the encoding of the character data on the DB2 source if it contains a mix of ASCII and EBCDIC data. Character set conversion by any given Replicat requires source data to be in a single character set.

The source character set is specified in the trail header. Thus, the Oracle GoldenGate trail can contain either ASCII or EBCDIC data, but not both. Unicode tables are

processed without any special configuration and are exempt from the one-character set requirement.

With respect to a source that contains both character encoding types, you have the following options:

- You can use one Extract for all of your tables, and have it write the character data to the trail as either ASCII or as EBCDIC.
- You can use different Extracts: one Extract to write the ASCII character data to a trail, and another Extract to write the EBCDIC character data to a different trail. You then associate each trail with its own data pump process and Replicat process, so that the two data streams are processed separately.

To output the correct character set in either of those scenarios, use the `TRAILCHARSETASCII` and `TRAILCHARSETEBCDIC` parameters. The default is `TRAILCHARSETEBCDIC`. Without these parameters, ASCII and EBCDIC data are written to the trail as-is. When using these parameters, note the following:

- If used on a single-byte DB2 subsystem, these parameters cause Extract to convert all of the character data to either the ASCII or EBCDIC single-byte CCSID of the subsystem to which Extract is connected, depending on which parameter is used (except for Unicode, which is processed as-is).
- If used on a multi-byte DB2 subsystem, these parameters cause Extract to capture only ASCII or EBCDIC tables (and Unicode). Character data is written in either the ASCII or EBCDIC mixed CCSID (depending on the parameter used) of the DB2 z/OS subsystem to which Extract is connected.

Specifying Multi-Byte Characters in Object Names

If the name of a schema, table, column, or stored procedure in a parameter file contains a multi-byte character, the name must be double-quoted. For more information about specifying object names, see *Administering Oracle GoldenGate*.

11

Preparing the DB2 for z/OS Transaction Logs for Oracle GoldenGate

Learn how to configure the DB2 transaction logging to support data capture by Oracle GoldenGate Extract.

Topics:

Preparing the DB2 z/OS Transaction Logs for Oracle GoldenGate

Learn to configure the DB2 transaction logging to support data capture by Oracle GoldenGate Extract.

Oracle GoldenGate can extract DB2 transaction data from the active and archived logs. Follow these guidelines to configure the logs so that Extract can capture data.

Topics:

Enabling Change Capture

Follow these steps to configure DB2 to log data changes in the expanded format that is supplied by the `DATA CAPTURE CHANGES` feature of the `CREATE TABLE` and `ALTER TABLE` commands. This format provides Oracle GoldenGate with the entire before and after images of rows that are changed with update statements.

1. From the Oracle GoldenGate directory, run GGSCI.
2. Log on to DB2 from GGSCI as a user that has `ALTER TABLE` privileges.

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

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

   ```
   ADD TRANDATA table
   ```

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

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

Enabling Access to Log Records

Activate DB2 Monitor Trace Class 1 ("`TRACE(MONITOR) CLASS(1)` ") so that DB2 allows Extract to read the active log. The default destination of `OPX` is sufficient, because Oracle GoldenGate does not use a destination.

To Start the Trace Manually

1. Log on to DB2 as a DB2 user who has the `TRACE` privilege or at least `SYSOPR` authority.
2. Issue the following command:

```
start trace(monitor) class(1) scope(group)
```

To Start the Trace Automatically When DB2 is Started

Do either of the following:

- Set `MONITOR TRACE` to "`YES`" on the `DSNTIPN` installation tracing panel.
- Set '`DSN6SYSP MON=YES` ' in the `DSNTIJUZ` installation job, as described in the *DB2 UDB Installation Guide*.

> **Note:**
>
> The primary authorization ID, or one of the secondary authorization IDs, of the ODBC plan executor also must have the `MONITOR2` privilege.

Sizing and Retaining the Logs

When tables are defined with `DATA CAPTURE CHANGES`, more data is logged than when they are defined with `DATA CAPTURE NONE` . If any of the following is true, you might need to increase the number and size of the active and archived logs.

- Your applications generate large amounts of DB2 data.
- Your applications have infrequent commits.
- You expect to stop Extract for long periods of time.
- Your network is unreliable or slow.

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

Retain enough log data so that Extract can start again from its checkpoints after you stop it or after an unplanned outage. Extract must have access to the log that contains the start of the oldest uncommitted unit of work, and all logs thereafter.

If data that Extract needs during processing was not retained, either in online or archived logs, one of the following corrective actions might be required:

- Alter Extract to capture from a later point in time for which log data is available (and accept possible data loss on the target).

- Resynchronize the source and target tables, and then start the Oracle GoldenGate environment over again.

> **Note:**
>
> The IBM documentation makes recommendations for improving the performance of log reads. In particular, you can use large log output buffers, large active logs, and make archives to disk.

Using Archive Logs on Tape

Oracle GoldenGate can read DB2 archive logs on tape, but it will degrade performance. For example, DB2 reserves taped archives for a single recovery task. Therefore, Extract would not be able to read an archive tape that is being used to recover a table until the recovery is finished. You could use DFHSM or an equivalent tools to move the archive logs in a seamless manner between online DASD storage and tape, but Extract will have to wait until the transfer is finished. Delays in Extract processing increase the latency between source and target data.

Controlling Log Flushes

When reading the transaction log, Extract does not process a transaction until it captures the commit record. If the commit record is on a data block that is not full, it cannot be captured until more log activity is generated to complete the block. The API that is used by Extract to read the logs only retrieves full physical data blocks.

A delay in receiving blocks that contain commits can cause latency between the source and target data. If the applications are not generating enough log records to fill a block, Extract generates its own log records by issuing `SAVEPOINT` and `COMMIT` statements, until the block fills up one way or the other and is released.

In a data sharing group, each API call causes DB2 to flush the data blocks of all active members, eliminating the need for Extract to perform flushes.

To prevent Extract from performing flushes, use the Extract parameter `TRANLOGOPTIONS` with the `NOFLUSH` option.

Part IV
Using Oracle GoldenGate for MySQL

Oracle GoldenGate for MySQL supports capture and delivery of initial load and transactional data for supported MySQL database versions.

Oracle GoldenGate for MySQL supports the mapping, filtering, and transformation of source data, unless noted otherwise in this document, along with replicating data derived from other source databases supported by Oracle GoldenGate, into MySQL databases.

This part describes tasks for configuring and running Oracle GoldenGate for MySQL and supported variants, such as MariaDB, Amazon RDS for MySQL, Amazon Aurora MySQL, Google Cloud SQL for MySQL, and SingleStoreDB.

Topics:

12

Understanding What's Supported for MySQL

This chapter contains information on database and table features supported by Oracle GoldenGate.

Topics:

Character Sets in MySQL

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

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

Limitations of Support

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

- Binary collations are not supported for multi-byte character sets. For example, do not set the collation_server variable equal to `utf8mb4_bin` when the character set is `utf8mb4`.
- The following character sets are not supported:

  ```
  armscii8
  utf8mb3
  keybcs2
  utf16le
  geostd8
  ```

Oracle GoldenGate for MySQL Supported Data Types

Oracle GoldenGate for MySQL supports the following data types:

- BLOB
- BIGINT
- BINARY
- BIT(M)
- CHAR
- DATE
- DATETIME
- DECIMAL
- DOUBLE
- ENUM
- FLOAT
- INT
- JSON
- LONGBLOB
- LONGTEXT
- MEDIUMBLOB
- MEDIUMINT
- MEDIUMTEXT
- SMALLINT
- TEXT
- TIME
- TIMESTAMP
- TINYBLOB
- TINYINT
- TINYTEXT
- VARBINARY
- VARCHAR
- YEAR
- Limitations and Clarifications

Limitations and Clarifications

When running Oracle GoldenGate for MySQL, be aware of the following:

- Functional indexes are not supported for Capture or Delivery.
- Oracle GoldenGate does not support `BLOB` or `TEXT` types when used as a primary key.
- Oracle GoldenGate supports a `TIME` type range from 00:00:00 to 23:59:59.
- Oracle GoldenGate supports timestamp data from `0001/01/03:00:00:00` to `9999/12/31:23:59:59`. If a timestamp is converted from GMT to local time, these limits also apply to the resulting timestamp. Depending on the time zone, conversion may add or subtract hours, which can cause the timestamp to exceed the lower or upper supported limit.
- Oracle GoldenGate does not support negative dates.
- The support of range and precision for floating-point numbers depends on the host machine. In general, the precision is accurate to 16 significant digits, but you should review the database documentation to determine the expected approximations. Oracle GoldenGate rounds or truncates values that exceed the supported precision.
- When you use `ENUM` type in non-strict `sql_mode`, the non-strict `sql_mode` does not prevent you from entering an invalid `ENUM` value and an error will be returned. To avoid this situation, do one of the following:
  - Use `sql_mode` as `STRICT` and restart Extract. This prevents users from entering invalid values for any of the data types. An IE user can only enter valid values for those data types.
  - Continue using non-strict `sql_mode`, but do not use `ENUM` data types.
  - Continue using non-strict `sql_mode` and use `ENUM` data types with valid values in the database. If you specify invalid values, the database will silently accept them and Extract will abend.
- Table with single column is not supported for JSON datatype. Extract will abend in case it is configured for a table which has a single column of `JSON` datatype.
- `JSON` datatype does not support CDR. The following message gets logged in the report file if `GETBEFORECOLS` is configured and the table has columns of `JSON` datatypes:

  ```
  INFO OGG-06556 The following columns will not be considered for CDR
  ```

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

Non-Supported MySQL Data Types

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

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

> **Note:**
>
> Extract abends if it is configured to capture from tables that contain any of the unsupported data types, so ensure that Extract is not configured to capture from tables containing columns of unsupported data types.

Supported Objects and Operations for MySQL

Oracle GoldenGate for MySQL supports the following objects and operations:

- Oracle GoldenGate supports the following DML operations on source and target database transactional tables:
  - Insert operation
  - Update operation (compressed included)
  - Delete operation (compressed included)
  - Truncate operation
- Oracle GoldenGate supports the extraction and replication of DDL (data definition language) operations.
- Oracle GoldenGate supports transactional tables up to the full row size and maximum number of columns that are supported by MySQL and the database storage engine that is being used. InnoDB supports up to 1017 columns.
- Generated columns are supported and captured.
- Oracle GoldenGate supports the `AUTO_INCREMENT` column attribute. The increment value is captured from the binary log by Extract and applied to the target table in a Replicat insert operation.
- Oracle GoldenGate can operate concurrently with MySQL native replication.
- Oracle GoldenGate supports the `DYNSQL` feature for MySQL.

> **Note:**
>
> XA transactions are not supported for capture and any XA transactions logged in `binlog` cause Extract to abend. So, you must not use XA transactions against a database that Extract is configured to capture. If XA transactions are being used for databases that are not configured for Oracle GoldenGate capture, then exclude those databases from logging into MySQL binary logs by using the parameter `binlog-ignore-db` in the MySQL server configuration file.

Limitations on Automatic Heartbeat Table support are as follows:

- Ensure that the database in which the heartbeat table is to be created already exists to avoid errors when adding the heartbeat table.
- In the heartbeat history lag view, the information in fields like `heartbeat_received_ts`, `incoming_heartbeat_age`, and `outgoing_heartbeat_age` are shown with respect to the system time. You should ensure that the operating system time is setup with the correct and current time zone information.

- Position by End of File (EOF) is supported in MySQL. Oracle GoldenGate Extract for MySQL finds the position corresponding to the end of the file and starts reading transactions from there. The EOF position is not exact, if data is continuously written to the binary log.

The Extract is added and altered using:

```
ADD EXTRACT group_name, TRANLOG, EOF


ALTER EXTRACT group_name, EOF
```

- Details of Support for Objects and Operations in MySQL DDL

Details of Support for Objects and Operations in MySQL DDL

Here's a list of the MySQL objects and operation types that Oracle GoldenGate supports for the capture and replication of DDL operations.

- DDL replication for MySQL is only supported between MySQL databases as sources and targets.
- Basic extraction and replication of DDL operations are supported for MySQL 5.7.10 and higher.
- For MySQL 5.7.10, only local DDL capture is supported.
- For MySQL 8.0, local and remote DDL capture is supported.
- Only the `CREATE TABLE`, `ALTER TABLE`, and `DROP TABLE` operations are supported.
- `TRUNCATE` operations are supported as DML through the `GETTRUNCATES` Extract and Replicat parameter and do not require configuring Oracle GoldenGate for MySQL DDL support.
- DDL replication is not supported in a Oracle GoldenGate bi-directional configuration.
- DDL replication is not supported for cloud based database services where the `binlog_row_metadata` database setting cannot be set to `FULL`.

Non-Supported Objects and Operations

Oracle GoldenGate for MySQL does not support the following objects and operations:

- Invisible columns
- The Oracle GoldenGate `BATCHSQL` feature.
- Array fetching during initial load.
- The following character sets are not supported:

  ULIB_CS_ARMSCII8, /* American National 166-9 */
  ULIB_CS_GEOSTD8, /* Geogian Standard */
  ULIB_CS_KEYBCS2, /* Kemennicky MS-DOS
- Capturing NLS LOB data using the `FETCHMOCOLS` and `FETCHMODCOLEXCEPT` TABLE options is not supported when DDL is enabled.
- Renaming tables.
- DDL statements inside stored procedures is not supported.
- When the time zone of the Oracle GoldenGate installation server does not match the time zone of the source database server, then the TIMESTAMP data sent to the target

database will differ from the source database. For Oracle GoldenGate Microservices installations, regardless of the time zones being the same, Extract will resolve the time zone to UTC. Determine the source database time zone by running the following query:

```
select @@system_time_zone;
```

This will return a time zone value, such as PDT.

For Classic Architecture, create a session variable for Oracle GoldenGate, called `TZ`, and set it equal to the time zone value of the database.

For MA, create a variable in the deployment that contains the source Extract, called `TZ` and set it to the value of the source database time zone. After this, stop any running Oracle GoldenGate processes and restart the Administration Service, and then start the Extracts and Replicats.

- Extraction and replication from and to views is not supported.
- Transactions applied by the slave are logged into the relay logs and not into the slave's `binlog`. If you want a slave to write transactions the `binlog` that it receives from the master , you need to start the replication slave with the log slave-updates option as 1 in `my.cnf`. This is in addition to the other binary logging parameters. After the master's transactions are in the slave's `binlog`, you can then setup a regular capture on the slave to capture and process the slave's `binlog`.

Limitations of SingleStoreDB Objects and Operations

Oracle GoldenGate for MySQL now supports SingleStoreDB and SingleStoreDB Cloud, but with the following limitations:

- Replication to views within SingleStoreDB are not supported, as only the base tables are writeable.
- Updates to columns that are part of the `SHARD` key are not supported.
- Primary key updates on tables with no explicit `SHARD` key are not supported. This is because SingleStoreDB assigns the primary key as the `SHARD` key in this situation, and updates to `SHARD` key columns are not allowed.
- The `DBOPTIONS LIMITROWS` behavior of Replicat for SingleStoreDB tables without a primary or unique key that are spread across multiple partitions, is not supported.
- The `DBOPTIONS LIMITROWS` and `NOLIMITROWS` parameter options are not supported for SingleStoreDB.
- SingleStoreDB does not support cross-database transactions, which means that a Replicat can only support mapping to a single schema/database. This includes mappings for checkpoint and heartbeat tables, so these objects must be created under the same schema/database as the user tables to be replicated.

Systems Schemas

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

- `'information_schema'`
- `'performance_schema'`

- `'mysql'`

13

Preparing and Configuring the System for Oracle GoldenGate

Learn about how to prepare the system for running Oracle GoldenGate and how to configure it with your MySQL database.

Topics:

Database User for Oracle GoldenGate Processes for MySQL

Requirements for the database user for Oracle GoldenGate processes are as follows:

- Create a database user that is dedicated to Oracle GoldenGate. It can be the same user for all the Oracle GoldenGate processes that must connect to a database.
- To support DDL replication, the MySQL user must have privileges to install the database plug-ins. The required permissions for the plug-in is only required with MySQL 5.7. The `INSERT` privilege is required on the `mysql.plugin` system table.
- To preserve the security of your data, and to monitor Oracle GoldenGate processing accurately, do not permit other users, applications, or processes to log on as, or operate as, the Oracle GoldenGate database user.
- Keep a record of the database users. They must be specified in the Oracle GoldenGate parameter files with the `USERID` parameter.
- The Oracle GoldenGate user requires read access to the `INFORMATION_SCHEMA` database.
- The Oracle GoldenGate user requires the following user privileges.

Privilege	Source Extract	Target Replicat	Purpose
`SELECT`	Yes	Yes	Connect to the database and select object definitions
`REPLICATION SLAVE`	Yes	NA	Connect and receive updates from the replication master's binary log.

`CREATE`
`CREATE VIEW`
`EVENT`
`INSERT`
`UPDATE`
`DELETE`
	Yes	Yes	Source and target database heartbeat and checkpoint table creation, and data record generation and purging

`DROP`	Yes	Yes	Dropping a Replicat checkpoint table or deleting a heartbeat table implementation
`EXECUTE`	Yes	Yes	To execute stored procedures.
`INSERT, UPDATE, DELETE` on target tables	NA	Yes	Apply replicated DML to target objects
DDL privileges on target objects (if using DDL support)	NA	Yes	Issue replicated DDL on target objects

- To capture binary log events, an Administrator must provide the following privileges to the Extract user:
  - Read and Execute permissions for the directory where the MySQL configuration file (`my.cnf`) is located
  - Read permission for the MySQL configuration file (`my.cnf`).

  - Read and Execute permissions for the directory where the binary logs are located.
  - Read and Execute permission for the `tmp` directory. The `tmp` directory is `/tmp`. The MySQL database connection requires access to the `/tmp/mysql.sock` file for versions prior to MySQL 8.0.

Ensuring Data Availability

Retain enough binary log data so that if you stop Extract or there is an unplanned outage, Extract can start again from its checkpoints. Extract must have access to the binary log that contains the start of the oldest uncommitted unit of work, and all binary logs thereafter. The recommended retention period is at least 24 hours worth of transaction data, including both active and archived information. You might need to do some testing to determine the best retention time given your data volume and business requirements.

If data that Extract needs during processing was not retained, either in active or backup logs, one of the following corrective actions might be required:

- Alter Extract to capture from a later point in time for which binary log data is available (and accept possible data loss on the target).
- Resynchronize the source and target tables, and then start the Oracle GoldenGate environment over again.

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

Setting Logging Parameters

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

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

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

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

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

1. Extract `TRANLOGOPTIONS` parameter with the `ALTLOGDEST` option. If this parameter specifies a location for the log index file, Extract accepts this location over any default that is specified in the MySQL Server configuration file. When `ALTLOGDEST` is used, the binary log index file must also be stored in the specified directory. This parameter should be used if the MySQL configuration file does not specify the full index file path name, specifies an incorrect location, or if there are multiple installations of MySQL on the same machine. From Oracle GoldenGate 21c onward, `ALTLOGDEST` parameter is optional for

local Extract, however, for remote Extract this parameter is mandatory. When ALTLOGDEST is not specified, the binary log index and binary log filepaths will be fetched from the database directly.Please note: The paths thus fetched are also subject to same accessibilitychecks as in the existing process.

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

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

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

```
TRANLOGOPTIONS ALTLOGDEST REMOTE
```

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

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

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

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

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

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

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

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

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

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

Database Connection

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

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

The `dbname` is the name of the MySQL instance, `hostname` is the name or IP address of the MySQL database server, `port` is the port number of the MySQL instance. If using an unqualified host name, that name must be properly configured in the DNS database. Otherwise, use the fully qualified host name, for example `myhost.company.com`.

Setting the Session Character Set

The `GGSCI`, Extract and Replicat processes use a session character set when connecting to the database. For MySQL, the session character set is taken from the `SESSIONCHARSET` option of `SOURCEDB` and `TARGETDB`. Make certain you specify a session character set in one of these ways when you configure Oracle GoldenGate.

Preparing Tables for Processing

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

Ensuring Row Uniqueness for Tables

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

Unless a `KEYCOLS` clause is used in the `TABLE` or `MAP` statement, Oracle GoldenGate selects a row identifier to use in the following order of priority:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. If none of the preceding key types exist (even though there might be other types of keys defined on the table) Oracle GoldenGate constructs a pseudo key of all columns that the database allows to be used in a unique key, excluding those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration.

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

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

Tables with a Primary Key Derived from a Unique Index

In the absence of a primary key on a table, MySQL will promote a unique index to primary key if the indexed column is `NOT NULL`. If there are more than one of these not-null indexes, the first one that was created becomes the primary key. To avoid Replicat errors, create these indexes in the same order on the source and target tables.

For example, assume that source and target tables named `ggvam.emp` each have columns named first, middle, and last, and all are defined as `NOT NULL`. If you create unique indexes in the following order, Oracle GoldenGate will abend on the target because the table definitions do not match.

Source:

```
mysql> create unique index uq1 on ggvam.emp(first);
mysql> create unique index uq2 on ggvam.emp(middle);
mysql> create unique index uq3 on ggvam.emp(last);
```

Target:

```
mysql> create unique index uq1 on ggvam.emp(last);
mysql> create unique index uq2 on ggvam.emp(first);
mysql> create unique index uq3 on ggvam.emp(middle);
```

The result of this sequence is that MySQL promotes the index on the source "first" column to primary key, and it promotes the index on the target "last" column to primary key. Oracle GoldenGate will select the primary keys as identifiers when it builds its metadata record, and the metadata will not match. To avoid this error, decide which column you want to promote to primary key, and create that index first on the source and target.

How to Specify Your Own Key for Oracle GoldenGate to Use

If a table does not have one of the preceding types of row identifiers, or if you prefer those identifiers not to be used, you can define a substitute key if the table has columns that always contain unique values. You define this substitute key by including a `KEYCOLS` clause within the Extract `TABLE` parameter and the Replicat `MAP` parameter. The specified key will override any existing primary or unique key that Oracle GoldenGate finds.

Limiting Row Changes in Tables That Do Not Have a Key

If a target table does not have a primary key or a unique key, duplicate rows can exist. In this case, Oracle GoldenGate could update or delete too many target rows, causing the source and target data to go out of synchronization without error messages to alert you. To limit the number of rows that are updated, use the `DBOPTIONS` parameter with the `LIMITROWS` option in the Replicat parameter file. `LIMITROWS` can increase the performance of Oracle GoldenGate on the target system because only one row is processed.

Triggers and Cascade Constraints Considerations

Triggers

Disable triggers on the target tables, or alter them to ignore changes made by the Oracle GoldenGate database user. Oracle GoldenGate replicates DML that results from a trigger. If the same trigger gets activated on the target table, then it becomes redundant because of the replicated version, and the database returns an error.

Cascade Constraints Considerations

Cascading updates and deletes captured by Oracle GoldenGate are not logged in binary log, so they are not captured. This is valid for both MySQL and MariaDB. For example, when you run the delete statement in the parent table with a parent child relationship between tables, the cascading deletes (if there are any) happens for child table, but they are not logged in binary log. Only the delete or update record for the parent table is logged in the binary log and captured by Oracle GoldenGate.

See https://mariadb.com/kb/en/replication-and-foreign-keys/ and https://dev.mysql.com/doc/refman/8.0/en/innodb-and-mysql-replication.html for details.

To properly handle replication of cascading operations, it is recommended to disable cascade deletes and updates on the source and code your application to explicitly delete or update the

child records prior to modifying the parent record. Alternatively, you must ensure that the target parent table has the same cascade constraints configured as the source parent table, but this could lead to an out-of-sync condition between source and target, especially in cases of bi-directional replication.

Changing the Log-Bin Location

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

1. Stop any new DML operations.
2. Let the extract finish processing all of the existing binary logs. You can verify this by noting when the checkpoint position reaches the offset of the last log.
3. After Extract finishes processing the data, stop the Extract group and, if necessary, back up the binary logs.
4. Stop the MySQL database.
5. Modify the `log-bin` path for the new location.
6. Start the MySQL database.
7. To clean the old log name entries from index file, use `flush master` or `reset master` (based on your MySQL version).
8. Start Extract.

Configure Bi-Directional Replication

In a bidirectional configuration, there are Extract and Replicat processes on both the source and target systems to support the replication of transactional changes on each system to the other system. To support this configuration, each Extract must be able to filter the transactions applied by the local Replicat, so that they are not recaptured and sent back to their source in a continuous loop. Additionally, tables whose key columns are `AUTO_INCREMENT` columns must be set so that there is no conflict between the values on each system.

1. Configure Oracle GoldenGate for high availability or active-active replication according to the instructions in the Propagating DDL in Active-Active (Bidirectional) Configurations in *Using Oracle GoldenGate for Oracle Database*.
2. To filter out Replicat operations in a bi-directional configuration so that the applied operations are not captured and looped back to the source again, take the following steps on each MySQL database:
   - Configure each Replicat process to use a checkpoint table. Replicat writes a checkpoint to this table at the end of each transaction. You can use one global checkpoint table or one per Replicat process See Overview of Replicat in *Understanding Oracle GoldenGate*.
   - Specify the name of the checkpoint table with the `FILTERTABLE` option of the `TRANLOGOPTIONS` parameter in the Extract parameter file. The Extract process will ignore transactions that end with an operation to the specified table, which should only be those of Replicat.

> **Note:**
>
> Although optional for other supported databases as a means of enhancing recovery, the use of a checkpoint table is required for MySQL when using bidirectional replication (and likewise, will enhance recovery).

If using a parallel Replicat in a bidirectional replication, then multiple filter tables are supported using the `TRANLOGOPTIONS FILTERTABLE` option. Multiple filter tables allow the `TRANLOGOPTIONS FILTERTABLE` to be specified multiple times with different table names or wildcards.

You can include single or multiple `TRANLOGOPTIONS FILTERTABLE` entries in the Extract parameter file. In the following example, multiple `TRANLOGOPTIONS FILTERTABLE`entries are included in the Extract parameter file with explicit object names and wildcards.

```
TRANLOGOPTIONS FILTERTABLE ggs.chkpt2
TRANLOGOPTIONS FILTERTABLE ggs.chkpt_RABC_*
```

3. If replicating data for tables that have `AUTO_INCREMENT` columns, edit the MySQL server and `auto_increment_offset` parameters to avoid discrepancies that could be caused by the bi-directional operations. The following illustrates these parameters, assuming two servers: **ServerA** and **ServerB**.

   ServerA:

   ```
   auto-increment-increment = 2
   auto-increment-offset = 1
   ```

   ServerB:

   ```
   auto-increment-increment = 2
   auto-increment-offset = 2
   ```

Configuring MySQL for Remote Capture

Oracle GoldenGate remote capture for MySQL, Amazon RDS for MySQL, Amazon Aurora MySQL, Azure Database for MySQL are used to capture transaction log data from a database located remotely to the Oracle GoldenGate installation.

Database Server Configuration

For remote capture to work, configure the MySQL server as follows:

1. Grant access permissions to the Oracle GoldenGate remote capture user.

   Run the following statements against the remote database to create the user and grant the permissions needed for remote capture.

   ```
   mysql > CREATE USER 'username'@'host' IDENTIFIED BY 'Password';
   mysql > GRANT ALL PRIVILEGES ON *.* TO 'username'@'host' WITH GRANT
   ```

```
OPTION;
mysql > FLUSH PRIVILEGES;
```

2. The `server_id` value of the remote MySQL server should be greater than 0. This value can be verified by issuing the following statement on the MySQL remote server:

   ```
   mysql > show variables like 'server_id';
   ```

   If the `server_id` value is 0, modify the `my.cnf` configuration file to set to a value greater than 0.

Oracle GoldenGate Configuration

Oracle GoldenGate configuration has the following steps:

1. Provide the remote database's connection information in the Extract's parameter file.

   ```
   SOURCEDB remotedb@mysqlserver.company.com:port, USERID username,
   PASSWORD
   password
   ```

2. Add the following parameter to the Extract's parameter file, after the connection information.

   ```
   TRANLOGOPTIONS ALTLOGDEST REMOTE
   ```

Limitations of Oracle GoldenGate Remote Capture for MySQL

Co-existence of Oracle GoldenGate for MySQL remote capture with the MySQL's native replication slave is supported with following conditions and limitations:

- The native replication slave should be assigned a different server_id than the currently running slaves. The slave server_id values can be seen using the following MySQL command on the master server.

  ```
  mysql> show slave hosts;
  ```

  - If the Oracle GoldenGate capture abends with error `"A slave with the same server_uuid or server_id as this slave has connected to the master"`, then change the capture's name and restart the capture.
  - If the native replication slave dies with the error `"A slave with the same server_uuid or server_id as this slave has connected to the master"`, then change the native replication slave's `server_id` and restart it.
- Remote capture is supported for Oracle GoldenGate on running on Linux and can support databases running on Linux or Windows.

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

To use the two way SSL in Oracle GoldenGate for MySQL capture and delivery, you need to supply the full paths of the certificate authority (`ca.pem`), the client certificate (`client-cert.pem`) and the client key (`client-key.pem`) files to the capture and delivery.

To know more about generating the certificate files, see:

https://dev.mysql.com/doc/refman/5.7/en/creating-ssl-rsa-files-using-mysql.html

You need to provide these paths in the Extract and Replicat parameter files using the `SETENV` parameter.

Following are the `SETENV` environment parameters to set the two-way SSL connection:

- `OGG_MYSQL_OPT_SSL_CA`: Sets the full path of the certification authority.
- `OGG_MYSQL_OPT_SSL_CERT`: Sets the full path of the client certificate.
- `OGG_MYSQL_OPT_SSL_KEY`: Sets the full path of the client key.

In the following example, the MySQL SSL certificate authority, client certificate, and client key paths are set to the Oracle GoldenGate MySQL Extract and Replicat parameter:

```
SETENV (OGG_MYSQL_OPT_SSL_CA='/var/lib/mysql.pem')
SETENV (OGG_MYSQL_OPT_SSL_CERT='/var/lib/mysql/client-cert.pem')
SETENV (OGG_MYSQL_OPT_SSL_KEY='/var/lib/mysql/client-key.pem')
```

For a MySQL user configured with X509 encryption scheme, the MySQL database requires the `ssl-key` and `ssl-cert` options at the time of logging in. So, when an Oracle GoldenGate credential store entry is created for this user, the SSL options in the credential store alias must mandatorily include `sslKey` and `sslCert` regardless of `sslMode` used.

Capturing using a MySQL Replication Slave

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

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

Other Oracle GoldenGate Parameters for MySQL

The following parameters may be of use in MySQL installations, and might be required if non-default settings are used for the MySQL database. Other Oracle GoldenGate parameters will be required in addition to these, depending on your intended business use and configuration.

Parameter	Description
DBOPTIONS with CONNECTIONPORT *port_number*	Required to specify to the VAM the TCP/IP connection port number of the MySQL instance to which an Oracle GoldenGate process must connect if MySQL is not running on the default of 3306. `DBOPTIONS CONNECTIONPORT 3307`
DBOPTIONS with HOST *host_id*	Specifies the DNS name or IP address of the system hosting MySQL to which Replicat must connect.
DBOPTIONS with ALLOWLOBDATATRUNCATE	Prevents Replicat from abending when replicated LOB data is too large for a target MySQL CHAR, VARCHAR, BINARY or VARBINARY column.

SOURCEDB with USERID and PASSWORD	Specifies database connection information consisting of the database, user name and password to use by an Oracle GoldenGate process that connects to a MySQL database. If MySQL is not running on the default port of 3306, you must specify a complete connection string that includes the port number: SOURCEDB *dbname@hostname:port*, USERID *user*, PASSWORD *password*.Example:
	`SOURCEDB mydb@mymachine:3307, USERID myuser, PASSWORD mypassword`
	If you are not running the MySQL database on port 3306, you must also specify the connection port of the MySQL database in the DBLOGIN command when issuing commands that affect the database through GGSCI:
	`DBLOGIN SOURCEDB dbname@hostname:port, USERID user, PASSWORD password`
	For example:
	`GGSCI> DBLOGIN SOURCEDB mydb@mymachine:3307, USERID myuser, PASSWORD mypassword`

SQLEXEC	To enable Replicat to bypass the MySQL connection timeout, configure the following command in a SQLEXEC statement in the Replicat parameter file. `SQLEXEC "select CURRENT_TIME();" EVERY n MINUTES` **Where**: *n* is the maximum interval after which you want Replicat to reconnect. The recommended connection timeout 31536000 seconds (365 days).

Parameter	Description

Global variable `sql_mode`	For heartbeattable to work in MySQL 5.7 , MySQL global variable `sql_mode` should not have `NO_ZERO_IN_DATE`, `NO_ZERO_DATE`. In the following example `sql_mode` includes `NO_ZERO_IN_DATE,NO_ZERO_DATE` values:
	
	`mysql> show variables like '%sql_mode%';+---------------+-+ + | Variable_name | Value`
	
	`| +--------------- +---------- - + | sql_mode | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_ DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION | +--------------- +---------- - ---------- + 1 row in set (0.00 sec)`
	
	These values must be removed by issuing the following command:
	
	`mysql> Set global sql_mode='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO _AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION'; Query OK, 0 rows affected, 1 warning (0.00 sec)`
	
	`mysql> show variables like '%sql_mode%'; +--------------- +---------- -+ | Variable_name | Value`
	
	`|`

Parameter	Description
	+-------------- +----------------------------------- --- -+ | sql_mode | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,ERROR_FOR _DIVISION_BY_ZERO,NO_AUTO_CREA TE_USER,NO_ENGINE_SUBSTITUTION | +-------------- +--------------------------------- --- -+ 1 row in set (0.01 sec)

Positioning Extract to a Specific Start Point

You can position the Extract to a specific start point in the transaction logs using the `ADD/ALTER EXTRACT` commands:

`{ADD | ALTER EXTRACT}` *`group`*`, LOGNUM` *`log_num`*`, LOGPOS` *`log_pos`*

- *`group`* is the name of the Oracle GoldenGate Extract group for which the start position is required.
- `LOGNUM` is the log file number. For example, if the required log file name is `test.000034`, the `LOGNUM` value is 34. Extract will search for this log file.

> **Note:**
>
> In Microservices Architecture, `ADD EXTRACT` will fail if the `LOGNUM` value contains zeroes preceding the value. For example, `ADD EXTRACT ext1, TRANLOG, LOGNUM 000001, LOGPOS 0` will fail. Instead, set `LOGNUM` to 1 for this example to succeed.

- *`LOGPOS`* is an event offset value within the log file that identifies a specific transaction record. Event offset values are stored in the header section of a log record. To position at the beginning of a `binlog` file, set the `LOGPOS` as 0.

In MySQL logs, an event offset value can be unique only within a given binary file. The combination of the position value and a log number will uniquely identify a transaction record. Maximum Log number length is 8 bytes unsigned integer and Maximum Log offset length is 8 bytes unsigned integer. Log number and Log offset are separated by a pipe (‘|’) delimiter. Transactional records available after this position within the specified log will be captured by Extract. In addition, you can position an Extract using a timestamp.

14

Using DDL Replication

Learn how to install, use, configure, and remove DDL replication.
Data Definition Language (DDL) statements (operations) are used to define MySQL database structures or schema. You can use these DDL statements for data replication between MySQL source and target databases. MySQL DDL specifics are found in the MySQL documentation at https://dev.mysql.com/doc/.

Oracle GoldenGate 21c for MySQL has introduced a transaction log based replication solution for MySQL 8.0, which has improved performance and usability when compared to the plug-in based DDL replication approach. However, plug-in based DDL replication approach is supported in older releases.

Topics:

Transaction Log Based DDL Configuration Prerequisites and Considerations

The prerequisites for configuring transaction log based DDL replication are as follows:

- Logging of full metadata is mandatory after upgrading to MySQL 8.0 onwards with Oracle GoldenGate 21c and higher. To enable full metadata logging:
  1. Set the value of MySQL server variable `binlog_row_metadata` to `FULL`, inside MySQL configuration file, `my.cnf` for Linux and `my.ini` for Windows.
  2. Restart the database server after changing the configuration file for the settings to take effect.
-
- DDL replication for remote capture is supported for MySQL 8.0 onwards. Transaction log based DDL replication works with both remote or local capture. This was a limitation for earlier Oracle GoldenGate releases. For example, Oracle GoldenGate 19c remote capture did not support DDL replication.
- Transaction log based DDL replication can handle DDLs issued within stored procedures, which is a limitation with plugin-based DDL replication.
- By design, the heartbeat table DDLs are ignored by the capture and you should create the heartbeat tables manually at the target.

Plug-in Based DDL Configuration Prerequisites and Considerations

This is an older approach to performing DDL Replication. The prerequisites for configuring DDL replication are as follows:

- DDL replication is supported for MySQL 5.7.
- Remote capture for MySQL 5.7 doesn't support DDL replication.
- Oracle GoldenGate DDL replication uses two plug-ins as a shared library, `ddl_rewriter` and `ddl_metadata`, which must be installed on your MySQL server before Oracle GoldenGate replication starts.
- The standalone application, Oracle GoldenGate `metadata_server`, must be running to capture the DDL metadata.
- The `history` table under the new `oggddl` database (`oggddl.history`). This metadata history table is used to store and retrieve the DDL metadata history. The history table records must be ignored from being logged into the binary log so you must specify `binlog-ignore-db=oggddl` in the `my.cnf` file.
- You should not manually drop the `oggddl` database or the `history` table because all DDL statements that run after this event will be lost.
- You should not stop the `metadata_server` during DDL capture as all the DDL statements that run after this event will be lost.
- You should not manually remove the `ddl_rewriter` and the `ddl_metadata` plugins during DDL capture because all DDL statements that run after this event will be lost.
- DDL executed within the stored procedure is *not* supported. For example , a DDL executed as in the following is *not* supported.

```
CREATE PROCEDURE atssrc.generate_data()
BEGIN
DECLARE i INT DEFAULT 0;
WHILE i < 800 DO
SET i = i + 1;
IF (i = 100) then
alter table atssrc.`ddl6` add col2 DATE after id;
ELSEIF (i = 200) then
alter table atssrc.`ddl6` add col3 DATETIME after datetime;
ELSEIF (i = 300) then
alter table atssrc.`ddl6` add `col4` timestamp NULL DEFAULT NULL
after
channel;
ELSEIF (i = 400) then
alter table atssrc.`ddl6` add col5 YEAR after value;
END IF;
END WHILE;
END$$
DELIMITER ;
call atssrc.generate_data();
```

- By design, the heartbeat table DDLs are ignored by the capture and you should create the heartbeat tables manually at the target.
- Installing DDL Replication
- Using the Metadata Server
- Troubleshooting Plug-in Based DDL Replication
- Upgrading from Plugin-based DDL Replication to Transaction Log-based DDL Replication
- Uninstalling Plug-In Based DDL Replication

Installing DDL Replication

To install DDL replication, you run the installation script that is provided with Oracle GoldenGate as the replication user. This user must have `Create`, `Insert`,`Select`, `Delete`, `Drop`, and `Truncate` database privileges. Additionally, this user must have write permission to copy the Oracle GoldenGate plugin in the MySQL plugin directory. For example, the MySQL plugin are typically in `/usr/lib64/mysql/plugin/`.

The installation script options are `install`, `uninstall`, `start`, `stop`, and `restart`.

The command to install DDL replication uses the install option, user id, password, and port number respectively:

```
bash-3.2$ ./ddl_install.sh install-option user-id password port-number
```

For example:

```
bash-3.2$ ./ddl_install.sh install root welcome 3306
```

The DDL replication installation script completes the following tasks:

1. Ensures that you have a supported MySQL server version installed. DDL replication is supported for MySQL 5.7.10 and greater.
2. Locates the MySQL plugin directory.
3. Ensures that the `ddl_rewriter`, `ddl_metadata` plugins and the `metadata_server` files exist. If these files are not found, then an error message appears and the installation exits.
4. Ensures that the plugins are already installed. If installed, the script exits with a message requesting you to uninstall first and then reinstall.
5. Stops the `metadata_server` if it is running.
6. Deletes the `oggddl.history` table if it exists.
7. Starts the `metadata_server` as a daemon process.
8. Installs the `ddl_rewriter` and `ddl_metadata` plugins.

Using the Metadata Server

You can use the following options with the metadata server:

- You must have the Oracle GoldenGate `metadata_server` running to capture the DDL metadata.
- Run the install script with `start` option to start the metadata server.
- Run the install script with `stop` option to stop the metadata server.
- Run the install script with `restart` option to stop the running metadata server and start again.
- Oracle GoldenGate DDL replication uses two plugins as a shared library, `ddl_rewriter` and `ddl_metadata`, both of which must be installed on your MySQL server before Oracle GoldenGate replication starts.
- The `oggddl.history` metadata history table is used to store and retrieve the DDL metadata history.

There is a single history table and metadata server for each MySQL server. If you want to issue and capture DDLs from multiple instances of an Extract process on the same database server at the same time, there is a possibility of conflict between accessing and populating the metadata history table. Oracle recommends that you do not run and capture DDLs using multiple Extract instances on the same MySQL server.

Troubleshooting Plug-in Based DDL Replication

Plug-in based DDL replication relies on a metadata history table and the metadata plugin and server. To troubleshoot when DDL replication is enabled, the history table contents and the metadata plugin server logs are required.
You can use the `mysqldump` command to generate the history table dump using one of the following examples:

```
mysqldump [options] database [tables]
mysqldump [options] --databases [options] DB1 [DB2 DB3...]
mysqldump [options] --all-databases [options]
```

For example, `bash-3.2$ mysqldump -uroot -pwelcome oggddl history > outfile`

The metadata plugins and server logs are located in the MySQL and Oracle GoldenGate installation directories respectively.

If you find an error in the log files, you need to ensure that the metadata server is running.

Upgrading from Plugin-based DDL Replication to Transaction Log-based DDL Replication

If you are using the plug-in based solution on MySQL 5.7 and plan to upgrade to MySQL 8.0, which uses transaction log based DDL replication, you need to:

1. Uninstall the plug-in components as mentioned in Uninstalling Plug-In Based DDL Replication
2. Upgrade your database.
3. Re-enable DDL replication support based on the steps provided in Transaction Log Based DDL Configuration Prerequisites and Considerations and check the prerequisites and configuration considerations.

Uninstalling Plug-In Based DDL Replication

If you no longer want to capture the DDL events, then you can use the same install script and select the `uninstall` option to disable the DDL setup. Also, any Extract with DDL parameters should be removed or disabled. If you want to capture the DDL again, you can run the install script again. You should take care when multiple instances of the capture process is running on the same instance of your MySQL server. The DDL setup should *not* be disturbed or uninstalled when multiple capture processes are running and when at most one capture is designed to capture the DDL statement.

Use the installation script with the `uninstall` option to uninstall DDL Replication. For example:

```
bash-3.2$ ./ddl_install.sh uninstall root welcome 3306
```

The script performs the following tasks:

1. Uninstalls the `ddl_rewriter` and `ddl_metadata` plugins.
2. Deletes the `oggddl.history` table if exists.
3. Removes the plugins from MySQL plugin directory.
4. Stops the `metadata_server` if it is running.

Using DDL Filtering for Replication

The following options are supported for MySQL DDL replication:

Option	Description
`DDL INCLUDE OPTYPE CREATE OBJTYPE TABLE;`	Include create table.
`DDL INCLUDE OBJNAME ggvam.*`	Include tables under the `ggvam`database.
`DDL EXCLUDE OBJNAME ggvam.emp*;`	Exclude all the tables under the `ggvam` database and table name starting with the `emp`wildcard.
`DDL INCLUDE INSTR 'XYZ'`	Include DDL that contains this string.
`DDL EXCLUDE INSTR 'WHY'`	Excludes DDL that contains this string.
`DDL INCLUDE MAPPED`	MySQL DDL uses this option and should be used as the default for Oracle GoldenGate MySQL DDL replication. `DDL INCLUDE ALL` and `DDL` are not supported.
`DDL EXCLUDE ALL`	Default option.

For a full list of options, see DDL in *Reference for Oracle GoldenGate*.

Using DDL Statements and Options

- `INCLUDE` (default) means include all objects that fit the rest of the description. `EXCLUDE` means to omit items that fit the description. Exclude rules take precedence over include rules.

- `OPTYPE` specifies the types of operations to be included or excluded. You can use `CREATE` and `ALTER`. Multiple `OPTYPE` can be specified using parentheses. For example, `optype (create, alter)`. The asterisk (*) wildcard can be specified to indicate all operation types, and this is the default.
- `OBJTYPE` specifies the `TABLE` operations to include or exclude. The wildcard can be specified to indicate all object types, and this is the default.
- `OBJNAME` specifies the actual object names to include or exclude. For example, `eric.*`. Wildcards are specified as in other cases where multiple tables are specified. The default is `*`.
- `String` indicates that the rule is true if any of the strings in `stringspec` are present (or false if `excludestring` is specified and the `stringspec` is present). If multiple `string` entries are made, at least one entry in each `stringspec` must be present to make the rule evaluate true.

  For example:

  ```
  ddlops string (“a”, “b”), string (“c”) evaluates true if string “a”
  OR “b” is present, AND string “c” is present
  ```

- `local` is specified if you want the rule to apply only to the current Extract trail (the Extract trail to which the rule applies must precede this `ddlops` specification).
- The semicolon is required to terminate the parameter entry.

  For example:

  ```
  ddl optype (create, drop), objname (eric.*);
  ddl exclude objname (eric.tab*);
  exttrail a;
  exttrail b;
  ddl optype (create), objname (joe.*), string (“abc”, “xyz”) local;
  ddl optype (alter), objtype (index);
  ```

  In this preceding example, the `exttrail a` gets creates and drops for all objects that belong to `eric`, except for objects that start with `tab`, `exttrail a` also gets all alter index statements, unless the index name begins with `tab` (the rule is global even though it’s included in `exttrail b`). `exttrail b` gets the same objects as `a`, and it also gets all creates for objects that belong to `joe` when the string `abc`or `xyz` is present in the DDL text. The `ddlops.c` module stores all DDL operation parameters and executes related rules.

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

See DDL and DDLOPTIONS.

15

Using Oracle GoldenGate with MySQL Group Replication

This topic describes the requirements and configuration steps for setting up Oracle GoldenGate to support MySQL Group Replication.

Topics:

Oracle GoldenGate Features to Support MySQL Group Replication

The following are Oracle GoldenGate features required to support capture from a MySQL database Group Replication instance.

CSN Format
The Extract for MySQL Group Replication uses a new CSN format that is based on the Group Replication Global Transaction ID. This CSN format should be used with `ATCSN` and `AFTERCSN` when manually positioning a MySQL Group Replication Extract or Replicat whose source trail was generated by a MySQL Group Replication Extract.
An example of the sequence used in group replication capture is:

```
00000000000000000001:f77024f9-f4e3-11eb-a052-0021f6e03f10:0000000000000010654
```

In this sequence, the Oracle GoldenGate sequence number is `00000000000000000001` and the GTID is `f77024f9-f4e3-11eb-a052-0021f6e03f10:0000000000000010654`.

Extended Checkpoint Support
The Extract for MySQL Group Replication includes an extended checkpoint file in addition to the core Extract checkpoint file. The extended checkpoint file is created in the same checkpoint directory where the core checkpoint and has a cpex extension after the name of the capture group for example, `extmysql.cpex`.
This file is created when Extract starts and is deleted when Extract is deleted and should not be edited.

Using GTID-based Extract
If `gtid_mode` is enabled in MySQL database, then Oracle GoldenGate Extract for MySQL automatically starts using the GTID-based recovery mechanism and extended checkpoint, which enables it to support failover and recovery. There is no extra parameter required for the Extract.

> **Note:**
>
> If not using Group Replication, it is recommended to disable `gtid_mode` on the source MySQL database. This will return the Extract's capture behavior to using the log number and offset method.

Requirements for Supporting Group Replication

This topic describes the requirements for using Oracle GoldenGate with MySQL Group Replication database clusters.

- Oracle GoldenGate for MySQL Group Replication supports MySQL version 8.0 and higher and requires Oracle GoldenGate version 21.7 or higher.
- Only Group Replication configured in Single-Primary Mode is supported for Extract.
- The MySQL database setting `gtid_mode` must be enabled.

Topic:

- Limitations of Group Replication with Oracle GoldenGate for MySQL

Limitations of Group Replication with Oracle GoldenGate for MySQL

The following limitations of support apply when using MySQL Group Replication with Oracle GoldenGate for MySQL:

- Group Replication configured for multi-primary mode is not supported.
- Extract does not support remote trails nor multiple local trails when configured against a MySQL Group Replication instance. If you need to use remote trails, then you can use data Pump to send the trail in Classic Architecture. In Microservices Architecture, use the DISTPATH to transport the trail. See Manage Distribution Paths.
- When Extract is running against a database configured for Group Replication, positioning by log number and offset is not supported.

SSL Configuration on Group Replication Cluster

Learn about SSL configuration on Group Replication Cluster.

Topics:

- Overview of Database Cluster SSL Configuration for Group Replication
- Create Server Certificates
- Configure Database Nodes and Router

Overview of Database Cluster SSL Configuration for Group Replication

A clustered database environment contains different nodes, constituting one primary node and one or more secondary nodes. There can be only one primary node at any instant. Each node has its own distinct hostname with a MySQL database instance, which is maintained by a separate configuration for that particular node. All the nodes in the cluster collectively represent the database.

There is a Router as well, which is the first point of contact for any client trying to connect to the database.

When enabling SSL connectivity, all of the database nodes and the Router will need to have their own authorization keys and server certificates. These certificates must be authorized by a common Certificate Authority (CA).

The certificates that are commonly used for this setup are:

- `ca.pem`: The certificate of the common CA (Certification Authority)
- `server-cert.pem`: The certificate that is certified by the CA for identifying the database node
- `server-key.pem`: The private key of the individual database node
- `router-cert.pem`: The certificate that is certified by the CA for identifying the router
- `router-key.pem`: The private key of the router

Configuration for the Router and database nodes is described in the following tables. For the purpose of this explanation, the following example shows one router and three database nodes.

Table 15-1 Router and Database Node Configuration

Router	**-**
Hostname	`mysqlrouter.company.com`
Config Filename	`mysqlrouter.conf`
Port	`6446`
Common Name	`mysqlrouter.company.com`
Certificate Name	`server-cert.pem`
Key file name	`server-key.pem`
Database Node 1	-
Hostname	`dbnode1.company.com`
Config Filename	`my.cnf`
Port	`3308`
Common Name	`dbnode1`
Certificate Name	`server-cert.pem`
Key file name	`server-key.pem`
Node Rank	Primary
Database Node2	-

Table 15-1 (Cont.) Router and Database Node Configuration

Router	-
Hostname	`dbnode2.company.com`
Config Filename	`my.cnf`
Port	`3308`
Common Name	`dbnode2`
Certificate Name	`server-cert.pem`
Key file name	`server-key.pem`
Node Rank	Secondary
Database Node3	-
Hostname	`dbnode3.company.com`
Config Filename	`my.cnf`
Port	`3308`
Common Name	`dbnode3`
Certificate Name	`server-cert.pem`
Key file name	`server-key.pem`
Node Rank	Secondary

Create Server Certificates

Before you begin configuring the router and database nodes, you'll need to create SSL server certificates. For connecting database nodes and router using SSL, you must have the right SSL keys and certificates for secure communication. All certificates must be recognized by a common Certification Authority (CA). If the keys and certificates were auto-generated during database/router installation (or if they are self-signed) then the connection might fail. Only certificates that are authorized by a CA are allowed to proceed further.

If the authorized server key and certificates are already available, then ensure that the certificates have the correct permissions and have been placed in the correct path for the router/database node.

For steps to generate SSL certificates for server, see:

Creating SSL Certificates and Keys Using OpenSSL

Tasks for Configuring SSL Certificates

1. Generate a separate certificate and key for each database node.
2. Use the same `ca.pem` which is common to all database nodes and routers.
3. In the server-certificate for the database nodes, specify the common name *without* the domain name. See the common name in the Table 15-1 in Overview of Database Cluster SSL Configuration for Group Replication for reference.
4. Ensure that the server **certificate name** and **key file name** match the corresponding database node and router values.

5. To verify the CN values in each generated server certificate, invoke openSSL using the following commands :

```
openssl x509 -text -in ca.pem
openssl x509 -text -in server-cert.pem
openssl x509 -text -in client-cert.pem
```

   The issuer CN must be the same for all. The subject CN must contain only hostname without domain name.

6. After generating the certificates, verify them against the CA file.
7. Copy the generated certificate and key file to the MySQL data directory for each database node and router. Ensure that you provide read permission to all users and retain write permission to file owner only.
8. Copy the common **`ca.pem`** to every node and router and provide read permissions to all users.

Configure Database Nodes and Router

Use the settings similar to the following, to configure database nodes and router for connecting over a secure SSL connection.

Router
In the Router config file, ensure that the below settings are present:

```
CLIENT_SSL_MODE=PREFERRED
CLIENT_SSL_CERT=absolute path of the generated router certificate
CLIENT_SSL_KEY=absolute path of the generated router key
SERVER_SSL_MODE=AS_CLIENT
SERVER_SSL_VERIFY=VERIFY_IDENTITY
SERVER_SSL_CA=absolute path of the common ca.pem placed on this server
```

After it is configured, provide read permissions to all users and revoke write permissions from group and others.

Database Node
In each of the MySQL database nodes, make sure the following are set under the appropriate section:

```
SSL_CAPATH=absolute path of the common ca.pem placed on this node
SSL_CA=ca.pem
SSL_CERT=server-cert.pem
SSL_KEY=server-key.pem
GROUP_REPLICATION_SSL_MODE=REQUIRED
REQUIRE_SECURE_TRANSPORT=ON
```

After configuring the database node, provide read permissions to all users and revoke write permissions from group and others.

Testing the Connection

After the configurations are in place and the appropriate permissions have been provided to the configuration files, test the settings by restarting the database nodes and router.

Test the Database Nodes Connection
Ensure that the database node does not terminate. Check the logs under log-error setting in the configuration file for any errors or warnings that indicate the SSL settings were not accepted. Try connecting to the specific node using the following command line (use the common name as specified in the certificate for this node):

```
mysql -u username -p password -h db_common_name -P db_port --ssl-
mode=VERIFY_IDENTITY --ssl-ca=path/of/ca.pem
```

Make sure that the connection does not generate any errors.
Similarly, connect with different SSL-modes by providing the appropriate parameter values.

> **Note:**
>
> The `ssl-cert` and `ssl-key` are not mandatory for `VERIFY_IDENTITY`. However, if the database user requires X509 authentication, then both `ssl-cert` and `ssl-key` must be provided with `client-cert` and `client-key`.

Test all database nodes using this method and then test the router connection.

Test the Router Connection
After the database nodes are up, restart the router and monitor it ensuring it does not terminate.
Check the logs under log-error setting in the configuration file for any errors or warnings that indicate the SSL settings were not accepted. If there are no errors or warnings, try connecting to the database from the router using the following command. Make sure you use the common name as specified in the certificate for the router:

```
mysql -u username -p password -h router_common_name -P router_port --
ssl-mode=VERIFY_IDENTITY --ssl-ca=path/of/ca.pem
```

Ensure that connection goes through without any errors.

Verify the Connection from the Router to the Database Node
First determine the currently active primary node, using the following command:

```
MySQL> SHOW VARIABLES like '%hosts%';
```

Now logout from the database and switchover the database to another node. Then login to the database from the router again, using the following command:

```
mysql -u username -p password -h router_common_name -P router_port --
ssl-mode=VERIFY_IDENTITY --ssl-ca=path/of/ca.pem
```

Check the currently active primary node using the same command again:

```
MySQL> SHOW VARIABLES like '%hosts%';
```

Part V
Using Oracle GoldenGate for PostgreSQL

Oracle GoldenGate for PostgreSQL supports capture and delivery of initial load and transactional data for supported PostgreSQL database versions.

Oracle GoldenGate for PostgreSQL supports the mapping, filtering, and transformation of source data, unless noted otherwise in this document, as well as replicating data derived from other source databases supported by Oracle GoldenGate, into PostgreSQL databases.

This part describes the tasks for configuring and running Oracle GoldenGate for PostgreSQL.

Topics:

16

Understanding What's Supported for PostgreSQL

This chapter contains information on supported features for Oracle GoldenGate for PostgreSQL:

Topics:

Supported Databases

The following are supported databases and limitations for Oracle GoldenGate for PostgreSQL:

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

Details of Supported PostgreSQL Data Types

This topic describes both supported and non-supported data types by Oracle GoldenGate for PostgreSQL:

Supported PostgreSQL Data Types

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

- `bigint`
- `bigserial`

- `bit(n)`
- `bit varying(n)`
- `boolean`
- `bytea`
- `char (n)`
- `cidr`
- `citext`
- `date`
- `decimal`
- `double precision`
- `inet`
- `integer`
- `interval`
- `json`
- `jsonb`
- `macaddr`
- `macaddr8`
- `money`
- `numeric`
- `real`
- `serial`
- `smallint`
- `smallserial`
- `text`
- `time` with/without timezone
- `timestamp` with/without timezone
- `uuid`
- `varchar(n)`
- `varbit`
- `xml`

Limitations of Support

- If columns of `char`, `varchar`, `text`, or `bytea` data types are part of a primary or unique key, then the maximum individual lengths for these columns must not exceed 8191 bytes.

- Columns of data type `CITEXT` that are part of the Primary Key are supported up to 8000 bytes in size. `CITEXT` columns that are greater than 8000 bytes and are part of the Primary Key are not supported.
- `real`, `double`, `numeric`, `decimal: NaN` input values are not supported.
- The following limitations apply to `bit`/`varbit` data types:
  - They are supported up to 4k in length. For lengths greater than 4k the data is truncated and only the lower 4k bits are captured.
  - The source bit(n) column can be applied only onto a character type column on a non-PostgreSQL target and can be applied onto a `char` type or a `bit/varbit` column on PostgreSQL target.
- The following limitations are applicable to both `timestamp` with time zone and `timestamp` without time zone:
  - The `timestamp` data with BC or AD tags in the data is not supported.
  - The `timestamp` data older than 1883-11-18 12:00:00 is not supported.
  - The `timestamp` data with more than 4 digits in the YEAR component is not supported.
  - `Infinity/-Infinity` input strings for `timestamp` columns are not supported.
- The following are the limitations when using `interval`:
  - The capture of mixed sign `interval` data from `interval` type columns is not supported. You can use `DBOPTIONS ALLOWNONSTANDARDINTERVALDATA` in the Extract parameter file to capture the mixed sign `interval` data (or any other format of `interval` data, which is not supported by Oracle GoldenGate) as a string (not as standard `interval` data).

    The following are a few examples of data that gets written to the trail file, on using the `DBOPTIONS ALLOWNONSTANDARDINTERVALDATA` in the Extract param file:
  - `+1026-9 +0 +0:0:22.000000` is interpreted as 1026 years, 9 months, 0 days, 0 hours, 0 minutes, 22 seconds.
  - `-0-0 -0 -8` is interpreted as 0 years, 0 months, 0 days, -8 hours.
  - `+1-3 +0 +3:20` is interpreted as 1 year, 3 months, 0 days, 3 hours, 20 minutes.
- Replicat: If the source interval data was captured using `DBOPTIONS ALLOWNONSTANDARDINTERVALDATA` and written as a string to the trail, the corresponding source column is allowed to be mapped to either a `char` or a `binary` type column on the target.
- `date` limitations are:
  - The `date` data with BC or AD tags in the data is not supported.
  - `Infinity`/`-Infinity` input strings for `date` columns are not supported.
- Columns of `text`, `json`, `xml`, `bytea`, `char (>8191)`, `varchar (>8191)` are treated as LOB columns and have the following limitations:
  - When using `GETUPDATEBEFORES`, the before image of LOB columns is never logged.
  - When using `NOCOMPRESSUPDATES`, `LOB` columns are logged in the after image only if they were modified.
- The support of range and precision for floating-point numbers depends on the host machine. In general, the precision is accurate to 16 significant digits, but you should

review the database documentation to determine the expected approximations. Oracle GoldenGate rounds or truncates values that exceed the supported precision.

Non-Supported PostgreSQL Data Types

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

- Arrays
- box
- circle
- Composite Types
- Domain Types
- Enumerated Types
- line
- lseq
- Object Identifiers Types
- path
- pg_lsn
- pg_snapshot
- point
- polygon
- Pseudo-Types
- Range Types
- tsquery
- tsvector
- User-defined Types (UDTs)
- Extensions and Additional Supplied Modules listed at: https://www.postgresql.org/docs/current/contrib.html

> **Note:**
>
> If the Extract parameter file contains a table with unsupported data types, the Extract will stop with an error message. To resume replication, remove the table from the Extract file or remove the column from the table with an unsupported data type.

> **Note:**
>
> If an Extension or Additional Supplied Module is supported, it will be explicitly added to the Supported PostgreSQL data types list.

Supported Objects and Operations for PostgreSQL

- Oracle GoldenGate for PostgreSQL only supports DML operations (Insert/Update/Deletes). DDL replication is not supported.
- Oracle GoldenGate for PostgreSQL supports replication of truncate operations beginning with PostgreSQL 11 and above, and requires the `GETTRUNCATES` parameter in Extract and Replicat.
- Case-Sensitive/Insensitive names Usage:
  - Unquoted names are case-insensitive and are implicitly lowercase. For example, `CREATE TABLE MixedCaseTable` and `SELECT * FROM mixedcasetable` are equivalent.
  - Quoted table and column names are case-sensitive and need to be listed correctly in Extracts and Replicats and with Oracle GoldenGate commands.

    For example, `TABLE appschema."MixedCaseTable"` and `ADD TRANDATA appschema."MixedCaseTable"` would be required to support a case-sensitive table name.
- Tables and Views
- Sequences and Identity Columns

Tables and Views

Tables to be included for capture and delivery must meet the following requirements and must only include data types listed under *Supported PostgreSQL Data Types*.

- Oracle GoldenGate for PostgreSQL supports capture of transactional DML from user tables, and delivery to user tables.
- Oracle GoldenGate for PostgreSQL supports delivery to partitioned tables.
- Globalization is supported for object names (table /schema/database/column names) and column data.

Limitations:

- Oracle GoldenGate for PostgreSQL does not support capture and delivery for views.
- Oracle GoldenGate for PostgreSQL does not support capture from partitioned tables.

Sequences and Identity Columns

- Sequences are supported on source and target tables for unidirectional, bidirectional, and multi- directional implementations.
- Identity columns created using the `GENERATED BY DEFAULT AS IDENTITY` clause are supported on source and target tables, for unidirectional, bidirectional, and multi-directional implementations.

- Identity columns created using the `GENERATED ALWAYS AS IDENTITY` clause are not supported in target database tables and the Identity property should be removed from target tables or changed to `GENERATED BY DEFAULT AS IDENTITY`.
- For bidirectional and multi-directional implementations, define the Identity columns and sequences with an `INCREMENT BY` value equal to the number of servers in the configuration, with a different `MINVALUE` for each one.

  For example, `MINVALUE` /`INCREMENT BY` values for a bidirectional, two-database configuration would be as follows:

  ```
  Database1, set the MINVALUE at 1 with an INCREMENT BY of 2.


  Database2, set the MINVALUE at 2 with an INCREMENT BY of 2.
  ```

  For example, `MINVALUE` /`INCREMENT BY` values for a multi-directional, three-database configuration would be as follows:

  ```
  Database1, set the MINVALUE at 1 with an INCREMENT BY of 3.


  Database2, set the MINVALUE at 2 with an INCREMENT BY of 3.


  Database3, set the MINVALUE at 3 with an INCREMENT BY of 3.
  ```

17

Preparing the Database for Oracle GoldenGate

This chapter describes how to prepare your PostgreSQL database and environment for Oracle GoldenGate.

Topics:

Database Configuration

To support Oracle GoldenGate, the following parameters in the PostgreSQL database configuration file, `$PGDATA/postgresql.conf`, needs to be configured.

- For remote connectivity of an Extract or Replicat, set the PostgreSQL `listen_addresses` to allow for remote database connectivity. For example:

  `listen_addresses=`*`remotehost_ip_address`*

  > **Note:**
  >
  > Ensure that client authentication is set to allow connections from an Oracle GoldenGate host by configuring the `pg_hba.conf` file. For more information, refer to this document: The pg_hba.conf File

- To support Oracle GoldenGate capture, **write-ahead logging** must be set to `logical`, which adds information necessary to support transactional record decoding.

  The number of **maximum replication slots** must be set to accommodate one open slot per Extract, and in general, no more than one Extract is needed per database. If for example PostgreSQL Native Replication is already in use and is using all of the currently configured replication slots, increase the value to allow for the registration of an Extract.

  Maximum **write-ahead senders** should be set to match the maximum replication slots value.

  Optionally, **commit timestamps** can be enabled in the write-ahead log, which when set at the same time logical write-ahead logging is enabled, will track the first DML commit record from that point on, with the correct timestamp value. Otherwise, the first record encountered by Oracle GoldenGate capture will have an incorrect commit timestamp.

```
wal_level = logical             # set to logical for Capture

max_replication_slots = 1       # max number of replication slots,
```

```
                                    # one slot per Extract/client

max_wal_senders = 1                 # one sender per max repl slot

track_commit_timestamp = on         # optional, correlates tx commit time
                                    # with begin tx log record (useful
for
                                    # timestamp-based positioning)
```

- After making any of the preceding changes, restart the database.
- Database Settings for PostgreSQL Cloud Databases

Database Settings for PostgreSQL Cloud Databases

Use these instructions to manage the database settings for Azure Database for PostgreSQL, Amazon Aurora PostgreSQL, Amazon RDS for PostgreSQL, and Google Cloud SQL for PostgreSQL.

Azure Database for PostgreSQL

When configuring Oracle GoldenGate for PostgreSQL Capture against an Azure Database for PostgreSQL, **logical decoding** must be enabled and set to `LOGICAL`.

Read the Microsoft Documentation for instructions.

Other database settings for Azure Database for PostgreSQL can be managed through the **Server parameters** section of the database instance.

For connections to an Azure Database for PostgreSQL instance, the default Azure Connection Security settings require SSL connections. To adhere to this requirement, further steps are required to support SSL connections with Oracle GoldenGate. Follow the content listed under Configuring SSL Support for PostgreSQL for more information.

Amazon Aurora PostgreSQL and Amazon RDS for PostgreSQL

For Amazon Aurora PostgreSQL and Amazon RDS for PostgreSQL, database settings are modified within parameter groups. Review the Amazon AWS documentation for information on how to edit database settings within a new parameter group and assign it to a database instance.

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_WorkingWithParamGroups.html

- Ensure that the database configuration settings listed previously are correct, by verifying them in the parameter group assigned to the instance.
- The `wal_level` setting for Amazon database services is configured with a parameter called `rds.logical_replication`, whose default is `0` and should be set to `1` if the database is to be used as source database for Oracle GoldenGate Capture.

Google Cloud SQL for PostgreSQL

When configuring an Oracle GoldenGate for PostgreSQL Extract for a Google Cloud SQL for PostgreSQL database, logical decoding must be set and is done by setting

the `cloudsql.logical_decoding` variable to `ON`. Follow the instructions provided by Google on how to enable this database flag. For more information, see https://cloud.google.com/sql/docs/postgres/flags#postgres-l.

Establishing Oracle GoldenGate Credentials

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

Topics:

Assigning Credentials to Oracle GoldenGate

Oracle GoldenGate processes require a database user to capture and deliver data to a PostgreSQL database and it is recommended to create a dedicated PostgreSQL database user for Extract and Replicat.

The following database user privileges are required for Oracle GoldenGate to capture from and apply to a PostgreSQL database.

Privilege	Extract	Replicat	Purpose
Database Replication Privileges			
`CONNECT`	Yes	Yes	Required for database connectivity. `GRANT CONNECT ON DATABASE dbname TO gguser;`
`WITH REPLICATION`	Yes	NA	Required for the user to register Extract with a replication slot. `ALTER USER gguser WITH REPLICATION;`
`WITH SUPERUSER`	Yes	NA	Required to enable table level supplemental logging (`ADD TRANDATA`) but can be revoked after TRANDATA is enabled for the table(s). `ALTER USER gguser WITH SUPERUSER;` For Azure Database for PostgreSQL, only the Admin user has SUPERUSER authority and is the only user that can enable TRANDATA.

Privilege	Extract	Replicat	Purpose
`USAGE ON SCHEMA`	Yes	Yes	For metadata access to tables in the schema to be replicated. `GRANT USAGE ON SCHEMA` *`tableschema`* `TO` *`gguser`*`;`
`SELECT ON TABLES`	Yes	Yes	Grant select access on tables to be replicated. `GRANT SELECT ON ALL TABLES IN SCHEMA` *`tableschema`* `TO` *`gguser`*`;`
`INSERT, UPDATE, DELETE, TRUNCATE` on target tables. Alternatively, if replicating every table, then you can use the `GRANT INSERT, UPDATE, DELETE, TRUNCATE ON ALL TABLES IN SCHEMA TO...` to the Replicat user, instead of granting `INSERT, UPDATE, DELETE` to every table.	NA	Yes	Apply replicated DML to target objects. `GRANT INSERT, UPDATE, DELETE, TRUNCATE ON TABLE` *`tablename`* `TO gguser;`
Heartbeat and Checkpoint Table Privileges			
`CREATE ON DATABASE`	Yes	Yes	Required by the Extract and Replicat user to add an Oracle GoldenGate schema for heartbeat and checkpoint table creation. `GRANT CREATE ON DATABASE` *`dbname`* `TO gguser;` Alternatively, if `GGSCHEMA` is the same as the user, then the objects can be created under the user by issuing `CREATE SCHEMA AUTHORIZATION` *`ggsuser`*`;`

Privilege	Extract	Replicat	Purpose
`CREATE, USAGE ON SCHEMA`	Yes	Yes	For heartbeat and checkpoint table creation/deletion if the Extract or Replicat user does not own the objects. `GRANT CREATE, USAGE ON SCHEMA ggschema TO gguser;`
`EXECUTE ON ALL FUNCTIONS`	Yes	Yes	For heartbeat update and purge function execution if the user calling the functions does not own the objects. `GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA ggschema TO gguser;`
`SELECT, INSERT, UPDATE, DELETE`	Yes	Yes	For heartbeat and checkpoint table inserts, updates and deletes if the user does not own the objects. `GRANT SELECT, INSERT, UPDATE, DELETE, ON ALL TABLES IN SCHEMA ggschema TO gguser;`

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.

Configuring a Database Connection

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

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

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

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

Topics:

Configuring a Database Connection in Linux

To create a database connection in Linux, set up a data source name (DSN) inside the `/etc/odbc.ini` file.

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

```
sudo vi /etc/odbc.ini

#Sample DSN entries
[ODBC Data Sources]
PG_src=DataDirect 7.1 PostgreSQL Wire Protocol
PG_tgt=DataDirect 7.1 PostgreSQL Wire Protocol


[ODBC]
IANAAppCodePage=4
InstallDir=/u01/app/ogg


[PG_src]
Driver=/u01/app/ogg/lib/GGpsql25.so
Description=DataDirect 7.1 PostgreSQL Wire Protocol
Database=sourcedb
```

```
HostName=remotehost
PortNumber=5432


[PG_tgt]
Driver=/u01/app/ogg/lib/GGpsql25.so
Description=DataDirect 7.1 PostgreSQL Wire Protocol
Database=targetdb
HostName=remotehost
PortNumber=5432
```

In the preceding examples:

`PG_src` and `PG_tgt` are user defined names of a source and target database DSN that will be referenced by Oracle GoldenGate processes, such as Extract or Replicat. DSN names are allowed up to 32 alpha-numeric characters in length, excluding special keyboard characters except for the underscore and dash.

`IANAAppCodePage=4` is the default setting but can be modified according to the following guidance, when the database character set is not Unicode.

https://docs.progress.com/bundle/datadirect-connect-odbc-71/page/IANAAppCodePage_9.html#IANAAppCodePage_9

`InstallDir` is the location of the Oracle GoldenGate installation folder.

`Driver` is the location of the Oracle GoldenGate installation home, `$OGG_HOME/lib/GGpsql25.so` file.

`Database` is the name of the source or target database.

`HostName` is the database host IP address or host name.

`PortNumber` is the listening port of the database.

You can also provide a `LogonID` and `Password` for the Extract or Replicat user, but these will be stored in clear text and it is recommended instead to leave these fields out of the DSN and instead store them in the Oracle GoldenGate wallet as a credential alias, and reference them with the `USERIDALIAS` parameter in Extract and Replicat.

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

Configuring a Database Connection on Windows

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

1. On the Windows system, open the **Control Panel** folder.
2. Open the **Administrative Tools** folder.
3. Open **ODBC Data Sources (64-bit).** The **ODBC Data Source Administrator** dialog box is displayed.
4. Select the **System DSN** tab, and then click **Add**.
5. Under **Create New Data Source**, select the **Oracle GoldenGate PostgreSQL Wire Protocol** driver and click **Finish**.
6. The **Create a New Data Source** wizard is displayed.
7. Supply the following:

- For **Data Source Name**, type a name for the DSN, up to 32 alpha-numeric characters in length, excluding special keyboard characters except for the underscore and dash.
- (Optional) For **Description**, type a description of this DSN.
- Provide the database server's **Host Name**, the database **Port Number**, and **Database Name**.

8. Click **OK** to close the dialog box.

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

Configuring SSL Support for PostgreSQL

SSL can be enabled by configuring the PostgreSQL configuration file (`$PGDATA/postgresql.conf`). For details, see Configuring SSL Support (PostgreSQL) in the *Securing the Oracle GoldenGate Environment*.

> **Note:**
>
> Azure Database for PostgreSQL defaults to enforce SSL connections. To adhere to this requirement, perform the requirements listed here, or optionally, you can disable enforcing SSL connections from the **Connection security** settings of the database instance using the Microsoft Azure Portal.

Preparing Tables for Processing

The following table attributes must be addressed in an Oracle GoldenGate environment for PostgreSQL.

Topics:

Disabling Triggers and Cascade Constraints on the Target

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

If not disabled, the same trigger or constraint gets activated on the target table and becomes redundant because of the replicated data. Consider the following example, where the source tables are `emp_src` and `salary_src` and the target tables are `emp_targ` and `salary_targ`

1. A delete is issued for `emp_src`.
2. It cascades a delete to `salary_src`.

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

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

Ensuring Row Uniqueness for Tables

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

Unless a `KEYCOLS` clause is used in the `TABLE` or `MAP` statement, Oracle GoldenGate selects a row identifier to use in the following order of priority:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. If none of the preceding key types exist (even though there might be other types of keys defined on the table) Oracle GoldenGate constructs a pseudo key of all columns that the database allows to be used in a unique key, excluding those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration. For PostgreSQL LOB types such as `text`, `json`, `xml`, `bytea`, `char`, `varchar`, Oracle GoldenGate supports these columns as a primary key in source or target tables up to a length of 8191 bytes.

   > **Note:**
   >
   > If there are other, non-usable keys on a table or if there are no keys at all on the table, Oracle GoldenGate logs an appropriate message to the report file. Constructing a key from all of the columns impedes the performance of Oracle GoldenGate on the source system. On the target, this key causes Replicat to use a larger, less efficient `WHERE` clause.
   > For tables that have no uniqueness and have repeat rows with the same values, Replicat will Abend on update and delete operations for these rows.

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

Enabling Table-Level Supplemental Logging

Enabling Supplemental logging is a process in which Oracle GoldenGate sets source database table level logging to support change data capture of source DML operations, and depending on the level of logging, to include additional, unchanged columns which would be needed in cases such as bi-directional replication with conflict detection and resolution configured.

There are four levels of table level logging in PostgreSQL, which equate to the `REPLICA IDENTITY` setting of a table, and those include `NOTHING`, `USING INDEX`, `DEFAULT`, and `FULL`.

Oracle GoldenGate requires `FULL` logging for use cases that require uncompressed trail records and Conflict Detection and Resolution, but in cases where tables have a Primary Key or Unique Index whose changes are being replicated in a simple uni-directional configuration or where full before-images or uncompressed records are not needed, then the `DEFAULT` level is acceptable. `NOTHING` and `USING INDEX` logging levels are not supported by Oracle GoldenGate and cannot be set with `ADD TRANDATA`.

Supplemental logging can be enabled within the Microservices Architecture web interface from the **Administration Service**, **Configuration** page, under the **Credential** created for a source database, or can be issued with the `ADD TRANDATA` command within GGSCI or the Microservices Architecture Admin Client.

The following is the syntax for issuing `ADD TRANDATA` from GGSCI.

```
GGSCI> DBLOGIN SOURCEDB dsn_name USERIDALIAS alias_name
 GGSCI> ADD TRANDATA schema.tablename ALLCOLS
```

> **Note:**
>
> For tables that have a primary key or unique index, the `ALLCOLS` option is required in order to set `FULL` logging for the table, otherwise `DEFAULT` logging is set.

`FULL` logging is always set for tables without a primary key or unique index, regardless of whether `ALLCOLS` is specified or not.

To check the level of supplemental logging:

```
GGSCI> INFO TRANDATA schema.tablename
```

18
Configuring Extract

This chapter contains instructions for configuring the Oracle GoldenGate Extract to capture initial load and transactional data from a PostgreSQL database.

Topics:

About Extract

For Oracle GoldenGate for PostgreSQL, there are two types of Extracts that can be created.

Initial Load Extract

An Initial Load Extract is used to read all records from a table and write them to an `EXTFILE` or `RMTFILE`. Initial load Extracts are created with the `SOURCEISTABLE` option of the `ADD EXTRACT` command and do not maintain checkpointing for recovery.

For more information on the Initial Load process, see Configuring an Initial Synchronization for a PostgreSQL Source Database using Precise Instantiation.

Change Data Capture Extract

A Change Data Capture Extract is used to capture transactional data changes from that point in time at which it is created or positioned into the write-ahead log.

The Oracle GoldenGate Extract process for PostgreSQL receives logical records from the PostgreSQL `test_decoding` database plugin and writes them in commit order into trail files for downstream consumption by a Replicat.

Extract Deployment Options

- **Local deployment**: For a local deployment, the source database and Oracle GoldenGate are installed on the same server. No extra consideration is needed for local deployments.
- **Remote deployment**: For a remote deployment, the source database and Oracle GoldenGate are installed on separate servers. Remote deployments are the only option available for supporting cloud databases, such as Azure for PostgreSQL or Amazon Aurora PostgreSQL.

For remote deployments, operating system endianness between the database server and Oracle GoldenGate server need to be the same, such as Windows and Windows, Linux and Linux, or Windows and Linux.

Server time and time zones of the Oracle GoldenGate server should be synchronized with that of the database server. If this is not possible, then positioning of an Extract when creating or altering one will need to be done by LSN.

In remote capture use cases, using `SQLEXEC` may introduce additional latency, as the `SQLEXEC` operation must be done serially for each record that the Extract processes. If special filtering that would require a `SQLEXEC` is done by a remote hub Extract and the performance impact is too severe, it may become necessary to move the Extract process closer to the source database.

With remote deployments, low network latency is important, and it is recommended that the network latency between the Oracle GoldenGate server and the source database server be less than 1 millisecond.

Prerequisites for Creating a Change Data Capture Extract

Review the Prerequisites for Installing Oracle GoldenGate for PostgreSQL.

Ensure that the database connection is configured correctly See Configuring a Database Connection for details.

- Registering the Extract for PostgreSQL

Registering the Extract for PostgreSQL

An Extract for PostgreSQL must be registered with the database and be granted a reserved replication slot. Replication slots are allocated through the database configuration setting `max_replication_slots` and can be configured as discussed in Database Configuration.

Follow these instructions to register an Extract. Extract registration must be done prior to creating an Extract. See `REGISTER EXTRACT` in the *Command Line Interface Reference for Oracle GoldenGate* for more information.

Topics:

- Registering Extract in Microservices Architecture for PostgreSQL
- Registering an Extract in Classic Architecture for PostgreSQL

Registering Extract in Microservices Architecture for PostgreSQL

1. Using the Admin Client, connect to the deployment, then connect to the credential alias for the source database.

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


OGG (https://remotehost:16000postgresql_source)> DBLOGIN USERIDALIAS alias
```

2. Register the Extract, which internally creates a replication slot for the Extract. Extract names cannot be more than 8 alpha-numeric characters.

```
OGG (https://remotehost:16000postgresql_source)> REGISTER EXTRACT extname
```

You can also register an Extract from the Oracle GoldenGate MA web interface. See How to Add Extracts in the *Step by Step Data Replication Using Oracle GoldenGate Microservices* guide.

Registering an Extract in Classic Architecture for PostgreSQL

Follow these instructions to register an Extract. Extract registration must be done prior to creating an Extract. See `REGISTER EXTRACT` in the *Command Line Interface Reference for Oracle GoldenGate* guide for more information.

1. Using GGSCI, connect to the DSN for the source database.

```
GGSCI> DBLOGIN SOURCEDB dsn USERIDALIAS alias
```

2. Register the Extract using the GGSCI command. This command internally creates the replication slot. Extract names cannot be more than 8 alpha-numeric characters.

```
GGSCI> REGISTER EXTRACT extname
```

Creating a Change Data Capture Extract

These steps configure a CDC Extract to capture transactional data from a source PostgreSQL database.

> **Note:**
>
> One Extract per database is generally sufficient, but multiple Extracts are allowed if the replication slots are available.

1. In GGSCI, Admin Client, or REST API client on the source system, create the Extract parameter file.

   ```
   EDIT PARAMS extname
   ```

   In this sample, *extname* is the name of the primary Extract and matches the name of the Extract that was registered with the database in the previous steps.

   To learn about using Oracle GoldenGate Microservices to perform this task, see How to Add Extracts in the *Step by Step Data Replication Using Oracle GoldenGate Microservices* guide.

2. Enter the Extract parameters in the order shown, starting a new line for each parameter statement.

   Sample basic parameters for Extract for Microservices installations:

   ```
   EXTRACT extname
   SOURCEDB dsn_name USERIDALIAS alias
   EXTTRAIL ep
   GETTRUNCATES
   TABLE schema.*;
   ```

   Sample basic parameters for Extract for Classic architecture installations:

   ```
   EXTRACT extname
   SOURCEDB dsn_name USERIDALIAS alias
   EXTTRAIL ./dirdat/ep
   GETTRUNCATES
   TABLE schema.object;
   ```

Parameter	Description
`EXTRACT extname`	*extname* is the name of the Extract and cannot be more than 8 alpha-numeric characters in length. For more information, see extract in *Reference for Oracle GoldenGate*.
`SOURCEDB dsn_name`	Specifies the name of the database connection DSN.
`USERIDALIAS alias`	Specifies the alias of the database login credential of the user that is assigned to Extract. This credential must exist in the Oracle GoldenGate credential store. For more information, see Establishing Oracle GoldenGate Credentials.
`EXTTRAIL trailname`	Specifies a two character, local trail to which the primary Extract writes captured data.
`GETTRUNCATES`	Optional parameter but needed in order to capture truncation operations.

Parameter	Description

`TABLE schema.object;`
or
`TABLE schema.*;`
	Specifies the database object for which to capture data.
	• `TABLE` specifies a table or a wildcarded set of tables.
	• `schema` is the schema name or a wildcarded set of schemas.
	• object is the table or sequence name, or a wildcarded set of those objects.
	• `*` is a wildcard for all tables in the schema.
	Terminate the parameter statement with a semi-colon.
	To exclude a name from a wildcard specification, use the `SCHEMAEXCLUDE`, `TABLEEXCLUDE`, and `EXCLUDEWILDCARDOBJECTSONLY` parameters as appropriate.
	Note:
	If the schema of tables to be captured from is the same as the schema in `GGSCHEMA` of the GLOBALS file, which is not recommended, then you cannot use `schema.*` in the `TABLE` statement.

3. Enter any optional Extract parameters that are recommended for your configuration. You can edit this file at any point before starting processing by using the `EDIT PARAMS` command.
4. Save and close the file.
5. Add the Extract and its associated trail file.

   For Microservices architecture using the Admin Client:

   ```
   OGG (https://remotehost:16000 postgresql_source)> ADD EXTRACT extname,
   TRANLOG, BEGIN NOW
   OGG (https://remotehost:16000 postgresql_source)> ADD EXTTRAIL ep,
   EXTRACT extname
   ```

   For Classic Architecture using GGSCI:

   ```
   GGSCI> ADD EXTRACT extname, TRANLOG, BEGIN NOW
   GGSCI> ADD EXTTRAIL ./dirdat/ep, EXTRACT extname
   ```

6. Start the Extract.

19

Configuring an Initial Synchronization for a PostgreSQL Source Database using Precise Instantiation

Data synchronization from a source PostgreSQL database to an Oracle GoldenGate target can be accomplished with the optional method of using precise instantiation. This method was introduced with Oracle GoldenGate 21c (21.8.0).

Precise instantiation has the advantage of not requiring any collision handling in the target Replicat. This is important for targets that do not support collision handling, such as flat files. This method uses a database snapshot to synchronize the output of the initial load Extract with the starting position of the Change Data Capture Extract. This snapshot is managed by the initial load Extract, so it is not possible for multiple initial load Extracts to use the same snapshot. Therefore, this method is not supported when using multiple intial load Extracts to parallelize the workload.

The following example uses the Admin Client within Microservices Architecture. It is assumed that you are familiar with Oracle GoldenGate and have setup the source and target databases correctly, with all required prerequisites. These steps require a minimum of Oracle GoldenGate 21c (21.8.0) or higher.

Perform the following steps to set up end-to-end initial load and synchronization processes using the precise instantiation method:

1. Register a Change Data Capture (CDC) Extract with the source PostgreSQL database.

   ```
   DBLOGIN USERIDALIAS src_alias
   REGISTER EXTRACT extecdc
   ```

   In this example, `extecdc` is the Extract name. For Classic Architecture installations, use the `DBLOGIN SOURCEDB` command with `USERIDALIAS` or `USERID` and `PASSWORD`.

2. Create an initial load Extract.

   ```
   ADD EXTRACT extinit, SOURCEISTABLE
   EDIT PARAMS extinit
   ```

   The initial load Extract parameter file must contain the `INITIALLOADOPTIONS USESNAPSHOT` parameter. For example:

   ```
   EXTRACT extinit
   INITIALLOADOPTIONS USESNAPSHOT
   SOURCEDB USERIDALIAS src_alias
   EXTFILE ./dirdat/ei, MEGABYTES 500, PURGE
   TABLE public.*;
   ```

See `INITIALLOADOPTIONS` to learn about the usage of this parameter with the `USESNAPSHOT` option.

3. Start the initial load Extract.

```
START EXTRACT extinit
```

4. When the initial load Extract has completed and stopped, review its report file to determine the positioning LSN to be used by the CDC Extract.

   For example, in the following output, the positioning LSN to be used by the CDC Extract will be '0/173F770'.

```
INFO    OGG-100001 A consistent point is
established in database 'tpcc' using replication slot
ogg_initx_1234 at LSN 0/173F770
and snapshot name '00000003-00000026-1'.


INFO    OGG-100002 Create or position the CDC
extract to LSN 0/173F770. Example: ADD EXTRACT <cdc-extract>
TRANLOG LSN 0/173F770
or ALTER EXTRACT <cdc-extract> LSN 0/173F770.
```

5. Create and start an initial load Replicat that reads the trail from the initial load Extract.

```
DBLOGIN USERIDALIAS tgt_alias


ADD CHECKTPOINTTABLE ggs.checkpoint


ADD REPLICAT repinit, EXTTRAIL ./dirdat/ei, CHECKPOINTTABLE
ggs.ggcheckpoint


START REPLICAT repinit
```

   Here is an example of the initial load Replicat parameter file:

```
REPLICAT repinit
TARGETDB USERIDALIAS tgt_alias
END RUNTIME
BATCHSQL
MAP public.*, TARGET public.*;
```

6. Add and start the CDC Extract (`extecdc`) using the consistent LSN value referred to in the initial load Extract report file.

```
ADD EXTRACT extecdc, TRANLOG, LSN 0/173F770


ADD EXTTRAIL ea, EXTRACT extecdc


START EXTRACT extecdc
```

   Here is an example of a CDC Extract parameter file:

```
EXTRACT extecdc
SOURCEDB USERIDALIAS src_alias
EXTTRAIL ./dirdat/ea
TABLE public.*;
```

7. When the initial load Replicat completes and stops, add and start a CDC Replicat that reads the trail from the CDC Extract.

```
ADD REPLICAT repecdc, EXTTRAIL ./dirdat/ea, CHECKPOINTTABLE
ggs.ggcheckpoint


START REPLICAT repecdc
```

8. Monitor the lag in both the CDC Extract and the CDC Replicat, and when they are both close to zero seconds, then the data stream from source to target database should be close to real-time.

20
Configuring Replicat

This chapter contains instructions for configuring the Replicat apply process to deliver data to a target PostgreSQL database.

Topics:

About Replicat

The Oracle GoldenGate Replicat for PostgreSQL reads data from Oracle GoldenGate source trail files and delivers the data to a target PostgreSQL database. The source trail data can be from any database that Oracle GoldenGate capture supports.

Available Replicats for PostgreSQL are Classic, Coordinated, and Parallel Replicat.

For more information on the differences between types of Replicats, review the Creating an Online Replicat Group content in the *Administering Oracle GoldenGate* guide.

Replicat Deployment Options

- **Local deployment**: For a local deployment, the target database and Oracle GoldenGate are installed on the same server. No extra consideration is needed for local deployments.
- **Remote deployment**: For a remote deployment, the target database and Oracle GoldenGate are installed on separate servers. Remote deployments are the only option available for supporting cloud databases, such as Azure for PostgreSQL or Amazon Aurora PostgreSQL.

  For remote deployments, operating system endianness between the database server and Oracle GoldenGate server needs to be the same, such as Windows and Windows, Linux and Linux, or Windows and Linux.

  With remote deployments, low network latency is important, and it is recommended that the network latency between the Oracle GoldenGate server and the target database server be less than 1 millisecond.

Prerequisites for Creating a Replicat

Review the Installing the DataDirect driver for PostgreSQL in *Installing Oracle GoldenGate* and ensure that the DataDirect drivers are installed correctly, which varies depending on the operating system.

Ensure that the PostgreSQL Client Authentication Configuration file, `$PGDATA/pg_hba.conf`, on the database server is configured to allow connections from the Oracle GoldenGate

server, if installed remotely. See https://www.postgresql.org/docs/13/auth-pg-hba-conf.html for more information.

- Creating a Checkpoint Table

Creating a Checkpoint Table

A checkpoint table is used by a Replicat in the target database for recovery positioning when restarting a Replicat. A checkpoint table is optional (but recommended) for a Classic Replicat and required for Coordinated and Parallel Replicats.
The checkpoint table needs to be created under an existing schema in the database and by default will attempt to create the table in the schema listed in the GLOBALS file, `GGSCHEMA` parameter. Ensure that the schema listed in GLOBALS exists in the database and that the Replicat user has permissions to use the schema and create tables in it. When creating a Microservices deployment, the schema that is bound to the GLOBALS file is the one entered in the **Replication Settings** step when creating the deployment.

These steps demonstrate creating a checkpoint table for a Classic and Coordinated Replicat. The checkpoint table for a Parallel Replicat is created when adding the Replicat and does not need to be created in advance.

- Microservices Architecture
- Classic Architecture

Microservices Architecture

1. Using the Admin Client, connect to the deployment, then connect to the credential alias for the target database.

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


OGG (https://remotehost:16000 postgresql_target)> DBLOGIN
USERIDALIAS alias
```

2. Add the checkpoint table:

```
OGG (https:// remotehost:16000postgresql_target)> ADD
CHECKPOINTTABLE
    ggadmin.oggcheck
```

   You can also add a checkpoint table from the Oracle GoldenGate MA Web UI. See Before Creating Replicat in the *Step by Step Data Replication Using Oracle GoldenGate Microservices* guide.

Classic Architecture

1. Using GGSCI, connect to the DSN for the target database.

```
GGSCI> DBLOGIN SOURCEDB dsn USERIDALIAS alias
```

2. Add the checkpoint table using the GGSCI command.

```
GGSCI> ADD CHECKPOINTTABLE ggadmin.oggcheck
```

Creating a Replicat

These steps create a Replicat to deliver transactional data to a target PostgreSQL database.

1. In GGSCI, Admin Client, or REST API client on the target system, create the Replicat parameter file.

```
EDIT PARAMS repnm
```

In this sample, `repnm` is a name of the Replicat. For classic Replicat, the name can be no more than 8 alpha-numeric characters in length. For Coordinated and Parallel Replicats, the name must be five or less alpha-numeric characters in length.

> **Note:**
>
> To learn about using Oracle GoldenGate Microservices to perform this task, see How to Add a Replicat in the *Step by Step Data Replication Using Oracle GoldenGate Microservices* guide.

2. Enter the Replicat parameters in the order shown, starting a new line for each parameter statement.

Sample basic parameters for classic Replicat and parallel Replicats:

```
REPLICAT repnm
TARGETDB dsn_name USERIDALIAS alias
BATCHSQL
GETTRUNCATES
MAP schema.object, TARGET schema.object;
```

Sample basic parameters for Coordinated Replicat:

```
REPLICAT repnm
TARGETDB dsn_name USERIDALIAS alias
BATCHSQL
GETTRUNCATES
MAP schema.object1, TARGET schema.object1, THREAD (1);
MAP schema.object2, TARGET schema.object2, THREAD (2);
MAP schema.object3, TARGET schema.object3, THREAD (3);
```

Parameter	Description
`REPLICAT repnm`	`repnm` is the name of the Replicat and cannot be more than 8 alpha-numeric characters in length for Classic Replicat and 5 or less for Coordinated and Parallel Replicats. For more information, see REPLICAT in *Reference for Oracle GoldenGate*.
`TARGETDB dsn_name`	Specifies the name of the database connection DSN.
`USERIDALIAS alias`	Specifies the alias of the database login credential of the user that is assigned to Replicat. This credential must exist in the Oracle GoldenGate credential store. For more information, see Establishing Oracle GoldenGate Credentials.
`BATCHSQL` `GETTRUNCATES`	Optional parameters for Replicat that supports transaction batching and replication of truncate operations.

`MAP schema.object, TARGET schema.object;`
or
`MAP schema.*, TARGET schema.*;`
	Specifies the relationship between a source table and the corresponding target object or objects.
	• `MAP` specifies the source table or a wildcarded set of tables.
	• `TARGET` specifies the target table or a wildcarded set of tables.
	• `schema` is the schema name or a wildcarded set of schemas.
	• `object` is the name of a table or a wildcarded set of tables.
	• `THREAD` assigns table operations to a specific coordinated Replicat thread.Terminate the parameter statement with a semi-colon.
	To exclude objects from a wildcard specification, use the `MAPEXCLUDE` parameter.
	For more information and for additional options that control data filtering, mapping, and manipulation, see `MAP` in *Reference for Oracle GoldenGate*.

3. 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.
4. Save and close the file.
5. Add the Replicat, which in this example, will be a Parallel Replicat.

   For Microservices Architecture using Admin Client:

   ```
   OGG (https://remotehost:16000postgresql_target)> ADD REPLICAT
   repnm, PARALLEL, EXTTRAIL ep, CHECKPOINTTABLE
   ggadmin.oggcheck
   ```

For Classic Architecture using GGSCI:

```
GGSCI> ADD REPLICAT repnm, PARALLEL, EXTTRAIL ./dirdat/ep,
CHECKPOINTTABLE ggadmin.oggcheck
```

6. Start the Replicat.

21
Additional Considerations

Adding a Heartbeat Table

Oracle GoldenGate for PostgreSQL supports a heartbeat table configuration, with some limitations regarding the update and purge tasks, which will be pointed out later. Adding a heartbeat table to both the source and target systems is optional but is useful in determining latency issues and to which process in the replication stream such issues may be occurring.

To add a heartbeat table for a database, review the required privileges in the Database Privileges for PostgreSQL and ensure that the correct database privileges are assigned to the Extract or Replicat user.

A schema in the database needs to be created and this should match the name of the schema used for the `GGSCHEMA` parameter of the GLOBALS file. The schema can be a unique schema that is not the same as the Extract or Replicat user, or can be the same as that user, but should always be reserved for Oracle GoldenGate objects only and should not be part of the user table schemas being replicated.

Microservices Architecture

1. Using the Admin Client, connect to the deployment, then connect to the credential alias for the source and target databases.

   ```
   OGG> CONNECT https://remotehost:srvmgrport DEPLOYMENT deployment_name AS
   deployment_user PASSWORD deployment_password
   OGG (https://remotehost:16000postgresql_target)> DBLOGIN USERIDALIAS alias
   ```

2. Add the heartbeat table:

   ```
   OGG (https:// remotehost:16000postgresql_source/target)> ADD
   HEARTBEATTABLE
   ```

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

You can also add a heartbeat table from the Oracle GoldenGate MA web interface. See Create a Heartbeat Table in the *Step by Step Data Replication Using Oracle GoldenGate Microservices* guide.

Classic Architecture

1. Using GGSCI, connect to the DSN for the source and target databases.

   ```
   GGSCI> DBLOGIN SOURCEDB dsn USERIDALIAS alias
   ```

2. Add the heartbeat table using the GGSCI command.

   ```
   GGSCI> ADD HEARTBEATTABLE
   ```

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

   ```
   GGSCI> ADD HEARTBEATTABLE TARGETONLY
   ```

To learn about heartbeat update and purge functions, see:

- Running the Heartbeat Update and Purge Function

Running the Heartbeat Update and Purge Function

The heartbeat table and associated functions are created from the `ADD HEARTBEATTABLE` command, however for PostgreSQL, there is no automatic scheduler to call the functions.

One main function controls both the heartbeat record update and the heartbeat history table purge functions. The default settings for both of these features are 60 seconds for the update frequency and 1 day for the history record purge, which deletes all records older than 30 days by default.

To call the main heartbeat record function, users should create an operating system level job that executes "`select ggschema.gg_hb_job_run();`". When this function is called, it will take into account the update frequency settings and history record purge settings and use those values regardless of the scheduler settings for the function call.

For example, users can create a **Cron Job** with the following syntax, and have it run every minute.

```
*****PGPASSWORD="gguserpasswd" psql -U gguser -d dbname -h remotehost -
p 5432 -c "select ggschema.gg_hb_job_run();" >/dev/null
 2>&1
```

Windows Task Scheduler, **pgAdmin**, or **pg_cron** are other programs that could be used to schedule the function call.

Enabling Bi-Directional Loop Detection

Loop detection is a requirement for bi-directional and multi-directional implementations of Oracle GoldenGate, so that an Extract for one source database does not recapture transactions sent by a Replicat from another source database.

With the CDC Extract capture method, by default, any transaction committed by a Replicat into a database where an Extract is configured, will recapture that transaction from the Replicat as long as supplemental logging is enabled for those tables that the Replicat is delivering to.

In order to ignore recapturing transactions that are applied by a Replicat, you must use the `TRANLOGOPTIONS FILTERTABLE` parameter for the CDC Extract. The table used as the filtering table will be the Oracle GoldenGate checkpoint table that you must create for the Replicat.

To create a Filter Table and enable Supplemental Logging:

1. On each source database, ensure that a checkpoint table for use by Replicats has been created. For example:

   ```
   ADD CHECKPOINTTABLE ggadmin.oggcheck
   ```

   It is recommended that you use the same schema name as used in the `GGSCHEMA` parameter of the `GLOBALS` file.

2. Enable supplemental logging for the checkpoint table. For example:

   ```
   ADD TRANDATA ggadmin.oggcheck ALLCOLS
   ```

3. Ensure that the Replicat is created with the checkpoint table information.

   ```
   ADD REPLICAT reptgt1, EXTTRAIL ./dirdat/e2, CHECKPOINTTABLE
   ggadmin.oggcheck
   ```

4. Configure each Extract with the `IGNOREREPLICATES` (on by default) and `FILTERTABLE` parameters, using the Replicat's checkpoint table for the filtering table.

   ```
   TRANLOGOPTIONS IGNOREREPLICATES
   TRANLOGOPTIONS FILTERTABLE ggadmin.oggcheck
   ```

> **Note:**
>
> Oracle GoldenGate for PostgreSQL supports only one `FILTERTABLE` statement per Extract, so for multi-directional implementations, ensure each Replicat uses the same checkpoint table in the database that they deliver to.

Deleting an Extract

When removing an individual Extract from use against a source PostgreSQL database, or uninstalling Oracle GoldenGate, the Extract that was registered with a replication slot in the database must be unregistered in order to remove the replication slot entry, otherwise an ever-increasing database log size can occur.

Perform the following steps to remove and unregister the Extract when no longer needed.

Microservices Architecture

1. Using the Admin Client, connect to the deployment, then connect to the credential alias for the source and target databases.

```
OGG> CONNECT https://remotehost:srvmgrport DEPLOYMENT
deployment_name AS deployment_user PASSWORD deployment_password
OGG (https://remotehost:16000postgresql_target)> DBLOGIN
USERIDALIAS alias
```

2. Delete the Extract first.

```
GGSCI> DELETE EXTRACT extname
```

3. Unregister the Extract.

```
GGSCI> UNREGISTER EXTRACT extname
```

Classic Architecture

1. Using GGSCI, connect to the DSN for the source and target databases.

```
GGSCI> DBLOGIN SOURCEDB dsn USERIDALIAS alias
```

2. Delete the Extract first.

```
GGSCI> DELETE EXTRACT extname
```

3. Unregister the Extract.

```
GGSCI> UNREGISTER EXTRACT extname
```

Removing Table-level Supplemental Logging

If a table is no longer required to be captured by Oracle GoldenGate and the `TABLE` parameter for the table has been removed from the Extract parameter file, or `TABLEEXCLUDE` is used to exclude the table from a wildcard statement, then supplemental logging can be removed from the table.

> **Note:**
>
> If the Extract resolves a table that does not have supplemental logging enabled, it will Abend depending on the type of DML operation.

Using `DELETE TRANDATA` to remove supplemental logging sets the Replicat Identity level of the table to `NOTHING`. Supplemental logging can be disabled using the Microservices Architecture web interface from the Administration Service, Configuration page, under the Credential created for a source database, or can be issued with the `DELETE TRANDATA` command within GGSCI or the Microservices Architecture Admin Client.

The following is the syntax for issuing `DELETE TRANDATA` from GGSCI.

```
GGSCI> DBLOGIN SOURCEDB dsn_name USERIDALIAS alias_name
 GGSCI> DELETE TRANDATA schema.tablename
```

To check the level of supplemental logging:

```
GGSCI> INFO TRANDATA schema.tablename
```

Part VI

Using Oracle GoldenGate for NonStop SQL/MX

Oracle GoldenGatefor NonStop SQL/MX supports data filtering, mapping, and transformation unless noted otherwise in this documentation. With the Oracle GoldenGate for NonStop SQL/MX databases, you can replicate data to and from supported NonStop SQL/MX versions or between a NonStop SQL/MX database and a database of another type.

This part describes tasks for configuring and running Oracle GoldenGate for NonStop SQL/MX.

Topics:

22

Understanding What's Supported for NonStop SQL/MX

This chapter contains information on database and table features supported by Oracle GoldenGate.

Topics:

Supported SQL/MX Data Types

The SQL/MX data types supported by Oracle GoldenGate are:

- `CHAR`
- `VARCHAR`
- `REAL`
- `DOUBLE`
- `NUMERIC`
- `SMALLINT`
- `LARGEINT`
- `DECIMAL`
- `VARCHAR(1)-(4040)`
- `FLOAT`
- `PIC`
- `DATE`
- `TIME`
- `TIMESTAMP`
- `SYSKEY`

Limitations of support:

- The original `SYSKEY` values are not preserved on the target. The target database generates a new unique value.
- Oracle GoldenGate does not support negative dates.
- Oracle GoldenGate supports timestamp data from 0001/01/03:00:00:00 to 9999/12/31:23:59:59. If a timestamp is converted from GMT to local time, these limits

also apply to the resulting timestamp. Depending on the time zone, conversion may add or subtract hours, which can cause the timestamp to exceed the lower or upper supported limit.

Supported Objects and Operations for SQL/MX

The objects and operations supported by Oracle GoldenGate for SQL/MX are:

- Oracle GoldenGate supports the extraction and replication of DML operations on tables that contain rows of up to 512 KB in length.
- Oracle GoldenGate supports the maximum number and size of columns per table that is supported by the database.
- Updates to primary keys are supported for SQL/MX version 3.2 and later.
- `PURGEDATA` operations
- SQL/MX now generates heartbeat records.

> **Note:**
>
> For other source GoldenGate versions higher than relelase 12.2, that send data to Oracle GoldenGate SQL/MX, the Extract on the source should use the option `FORMAT RELEASE 12.2` for `EXTTRAIL/RMTTRAIL` parameter.

Limitations on Automatic Heartbeat Table support are as follows:

- The GLOBALS heartbeat table only supports two-part and three-part names. You can use `ENABLECATALOGNAMES` to enable these table names.
- Do not use the Oracle GoldenGate default schema, `GGSCHEMA`; you must use a two or three-part name only.

Non-Supported Objects and Operations for SQL/MX

The objects and operations that Oracle GoldenGate does not support for SQL/MX are:

- Extraction or replication of DDL (data definition language) operations
- Oracle GoldenGate `SQLEXEC` functionality not supported in Extract.
- Oracle GoldenGate parameters that involve fetching from the database, such as `FETCHCOLS`, `FETCHCOLSEXCEPT`, and `FETCHBEFOREFILTER`.
- NonStop SQL/MX distributed transactions
- Views
- Datatypes
  - `BINARY`
  - `VARBINARY`

23

Preparing the Database for Oracle GoldenGate

This chapter describes how to prepare your NonStop SQL/MX database and environment for Oracle GoldenGate.

Topics:

Database User for Oracle GoldenGate Processes

Create a database user that is dedicated to Oracle GoldenGate. It can be the same user for all of the Oracle GoldenGate processes that must connect to a database:

- Extract (source SQL/MX database)
- Replicat (target SQL/MX database)
- `DEFGEN` (source or target database)

Table 23-1 Database user for Oracle GoldenGate processes

Privilege	Extract user	Replicat user	`DEFGEN` user
`SELECT`	X	X	X
`DELETE`	X		
`INSERT`	X		
`UPDATE`	X		
`REFERENCES`	X		

SQL/MX Access Privileges

Dedicate an NSK user (`groupID.userID`) or OSS alias userID to Oracle GoldenGate. This user requires the following access privileges at the SQL/MX data level:

- table
- view
- stored procedure.

Access privileges are granted through the SQL/MX command interface with a *GRANT* statement. For more information on the `GRANT` command, see the SQL/MX documentation.

24
Preparing the System for Oracle GoldenGate

This chapter contains guidelines for preparing the NonStop SQL/MX system to support Oracle GoldenGate:

Topics:

Configuring Oracle GoldenGate to Support Tables with a `SYSKEY` or Clustered Key

If your Replicat configuration includes tables that have a `SYSKEY` or a clustered key, you must take steps to ensure that the correct row is changed during an `UPDATE` or `DELETE` operation. Replicat uses an ODBC driver to connect to a target SQL/MX database. This driver does not allow Replicat to get the values of the `SYSKEYs` in the target tables, so they cannot be used in the `WHERE` clause to locate rows for processing. You can handle this condition in either of the following ways:

Topics:

Method One (Handle Using Your Replicat Configuration)

> **Note:**
>
> This is the preferred method of handling tables with `SYSKEY` or clustered keys.

This method maps the source `SYSKEY` to an additional column in the target table. Replicat can then be configured to use the key value to locate a target row that has the same key value.

1. Add a column named `GGS_SYSKEY` to the target table.

2. In the `MAP` statement, map the source `SYSKEY` column to the `GGS_SYSKEY` column by means of a `COLMAP` clause.
3. Specify the `GGS_SYSKEY` column in a `KEYCOLS` clause in the same `MAP` statement. This ensures that the unique source `SYSKEY` value is used as the key for the target table.

The following is an example of this procedure:

Source table:

```
CREATE TABLE DEV.TSSCAT.ENTRY
  (
    COLA    INT DEFAULT NULL
  , COLB    CHAR(20)  DEFAULT DEFAULT NULL
  )
```

Target table:

If the target table has a clustered key rather than a `SYSKEY`, include the user-defined columns in the `STORE BY` clause.

```
CREATE TABLE DEV.TASCAT.ENTRY
  (
    GGS_SYSKEY  LARGEINT NO DEFAULT NOT NULL
  , COLA        INT DEFAULT NULL
  , COLB        CHAR(20) DEFAULT DEFAULT NULL
  )
  STORE BY (GGS_SYSKEY ASC);
```

MAP statement:

If the target table has a clustered key rather than a `SYSKEY`, include the user-defined columns in the `STORE BY` clause.

```
MAP TSSCAT.ENTRY, TARGET TASCAT.ENTRY,
   COLMAP (ggs_syskey = syskey, USEDEFAULTS),
   KEYCOLS (ggs_syskey);
```

Method Two (Handle Using Your Extract Configuration)

You can allow Replicat to use the full row image to locate a row in the target table, rather than rely on a key. Replicat defaults to using the full row image when a target key is not defined or when it is not available (as in this case, where ODBC does not supply it). This method requires that the combination of all of the column values of any given row makes that row unique among all other rows in the table. Otherwise, Replicat may change more than one row.

No changes to the target table definitions are required by this method, nor are there any special column mapping requirements in the Replicat configuration. You only need to turn off compression in the source table attributes and in the Extract configuration.

1. Create or alter the source tables to have the `ATTRIBUTE NO AUDITCOMPRESS` set. The following is an example:

   ```
   CREATE TABLE DEV.TSSCAT.ENTRY
     (
       COLA    INT DEFAULT NULL
     , COLB    CHAR(20)  DEFAULT DEFAULT NULL
   ```

```
)
ATTRIBUTE NO AUDITCOMPRESS;
```

2. Use the `NOCOMPRESSDELETES` and `NOCOMPRESSUPDATES` parameters in the Extract parameter file to configure Extract to write all of the columns of a table to the trail for `UPDATE` and `DELETE` operations. Replicat will use all column values as the row locator. (By default Extract only writes the key to the trail for `DELETEs` and only the key and the changed columns for `UPDATEs`.) For more information, see *Reference for Oracle GoldenGate*.

Define an ODBC data source

Follow these steps to specify a data source name (DSN) to which the GGSCI command interface can connect on the source and to which the Replicat process can connect on the target.

1. Log into the NonStop system and select a TACL prompt.
2. Edit or TEdit the `$SYSTEM.SYSTEM.ODBCDSN` ODBC configuration file.
3. Add the DSN to the `[ODBC Data Sources]` list, as shown in Example 24-1. A DSN of `TDM_Default_DataSource` and default connection settings are included in this file by default.
4. Define your data source connection by adding the following lines as needed:
   - [*dsn*]: Replace *dsn* in the heading with the actual DSN.
   - `Description`: Add a text string description, if needed.
   - `Catalog`: Add the database catalog.
   - `Schema`: Add the database schema.
   - `Server`: Add the NSK server. The server is where the ODBC/MX server is running and must be in the format of `TCP:`*IP_address_or_domain_name*`/`*IP_port* as shown in Example 24-1.
   - Add the other parameters only if you want them to be something other than the default settings specified under `TDM_Default_DataSource` .

> **Note:**
>
> The DSN in the `ODBCDSN` file must exactly match the DSN that is defined in the `ODBC/MX` service. Data source names are case-sensitive.

5. Save the file and then exit the edit session.

Example 24-1 Template for ODBC configuration file

```
TACL> Edit $SYSTEM.SYSTEM.ODBCDSN

[ODBC]
TraceFlags = 6
TraceStart = 0TraceFile = trlog

[ODBC Data Sources]
TDM_Default_DataSource = NonStop ODBC/MX 2.3
<dsn> = NonStop ODBC/MX 2.3
```

```
DataSourceName = <Driver>

[TDM_Default_DataSource]
Description = Default Data Source
Catalog = CAT
Schema = SCH
DataLang = 0
FetchBufferSize = SYSTEM_DEFAULT
Server = TCP:xxx.xxx.xxx.xxx/xxxx
SQL_ATTR_CONNECTION_TIMEOUT = NO_TIMEOUT
SQL_LOGIN_TIMEOUT = NO_TIMEOUT
SQL_QUERY_TIMEOUT = NO_TIMEOUT

[<dsn>]
Description = <text string describing data source>
Catalog = <target catalog>
Schema = <target schema>
Server = TCP:<ip address or domain name>/<ip port>
```

For more information about the `$SYSTEM.SYSTEM.ODBCDSN` file and how to configure ODBC for SQL/MX, see *HP NonStop Open System Services ODBC/MX Client Driver* at.

https://support.hpe.com/hpsc/doc/public/display?docId=emr_na-c02132824

Specifying Object Names in a SQL/MX Configuration

Oracle GoldenGate supports both two-part (`schema.table`) and three-part (`catalog.schema.table`) table names in parameter files and commands. See Specifying Object Names in Oracle GoldenGate Input in *Administering Oracle GoldenGate*for more information.

- Using Two-Part Names
- Using Three-Part Names

Using Two-Part Names

In a two-part name, you specify the schema and the object name. To map the catalog portion of the name, you must link it to an Extract or Replicat group. Only one catalog can be linked to an Extract or Replicat group. To capture from, or apply to, more than one catalog, you must create an Extract group for each one on the source and a Replicat group for each one on the target. To link a catalog to an Extract group, use the `SOURCEDB` parameter. To link a catalog to a Replicat group, use the `TARGETDB` parameter. The `USERID` portion of `SOURCEDB` and `TARGETDB` specifies the default schema.

```
SOURCEDB catalog USERID schema

TARGETDB catalog USERID schema
```

> **Note:**
>
> The API that is used by Extract does not log in to the database, so an authentication password is not required.

Using Three-Part Names

In a three-part name, you specify the catalog, schema, and object name as `catalog.schema.object`. You must explicitly enable support for three-part names by using the `ENABLECATALOGNAMES` parameter in the `GLOBALS` file. For more information about the `GLOBALS` file, see *Administering Oracle GoldenGate*.

Disable Triggers and Cascade Constraints on a SQL/MX Target

Disable triggers, cascade delete constraints, and cascade update constraints on target SQL/MX tables, or alter them to ignore changes made by the Oracle GoldenGate database user. Oracle GoldenGate replicates DML that results from a trigger or cascade constraint. If the same trigger or constraint gets activated on the target table, it becomes redundant because of the replicated version, and the database returns an error. Consider the following example, where the source tables are `emp_src` and `salary_src` and the target tables are `emp_targ` and `salary_targ`.

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

Configure ODBC to Prevent Timeouts on a SQL/MX Target

Follow this procedure to change the ODBC connection timeout on a NonStop target from the `SYSTEM_DEFAULT` of ten minutes to `NO_TIMEOUT` .

1. From OSH, run `mxci` and set the mode to `mxcs` .

   ```
   /G/DEV01/SUPERDEV 1>mxci
   >>mode mxcs;
   ```

2. Issue the following command to show current settings.

   ```
   info ds *,detail;
   ```

   The system responds:

3. Change the `IdleTimeout` and `ConnTimeout` to `NO_TIMEOUT` as follows:

   ```
   CS>alter ds "TDM_Default_DataSource", IdleTimeout NO_TIMEOUT;
   CS>>alter ds "TDM_Default_DataSource", ConnTimeout NO_TIMEOUT;
   ```

4. Repeat the prior alters for any `ds` Oracle GoldenGate might also use.

Example 24-2

```
Name: \SYSA.$MX.TDM_Default_DataSource
CpuList: ALL
InitPri.........Same as Assoc Server
CurrentState.................STARTED
ConnectedServers...................0
AvailableServers...................4
LastStateChg.............Apr 12 15:36
LastUpdate..............Apr 12 14:56
IdleServer.........................4
IdleTimeout............SYSTEM_DEFAULT
MaxServer.........................100
ConnTimeout............SYSTEM_DEFAULT
InitServer.........................4
StartAutomatic.....................ON
Trace............................OFF
SQLPrepareStat....................OFF
ConnInfoStat.......................ON
SQLExecuteStat....................OFF
SessionInfoStat....................ON
SQLExecDirectStat.................OFF
SQLStmtStat.......................OFF
SQLFetchStat......................OFF
```

Specify Connection Authentication on a SQL/MX Target

Add the following parameters to the Replicat parameter file to specify ODBC connection authentication for Replicat to use on the target SQL/MX database.

`TARGETDB` *`DSN`* `USERID` *`user`*`,` `PASSWORD` *`password`*

- Use the `TARGETDB` and `USERID` parameters as one entry.
- Supply the ODBC data source name with `TARGETDB` .
- Supply the user name and password with `USERID` .

Supply a Data Definitions File on a NonStop Target

To replicate data between source and target NonStop SQL/MX databases, you must supply source data definitions to the Replicat process, even though the two databases might be identical in version and structure. There are slight differences in the way that metadata is returned to Oracle GoldenGate by the native API from the source database and by ODBC from the target database.

1. Create a data definitions file with the `DEFGEN` utility.
2. Transfer the definitions file to the target system.
3. Specify the fully qualified name of the definitions file with the `SOURCEDEFS` parameter in the Replicat parameter file.

For more information about data-definitions files, see Understanding Data Definition Files.

25

Configuring Manager and Other Processes

To configure Oracle GoldenGate to support your business requirements, see Getting Started with the Oracle GoldenGate Process Interfaces. It contains instructions to:

- Configure the Manager process with a TCP/IP port and other optional parameters that control dynamic port assignments, trail file maintenance, automatic startup, and other properties.
- Configure Extract and Replicat processes to support reporting, high availability, disaster recovery, and other topologies.
- Configure security to control user access, file security, and data encryption.
- Configure integration, manipulation, and conversion features that enable you to customize Oracle GoldenGate and support the delivery of data across heterogeneous environments.
- Configure utilities and other tools that support Oracle GoldenGate.

Part VII
Using Oracle GoldenGate for SQL Server

With Oracle GoldenGate for SQL Server supports capture and delivery of initial load and transactional data for supported SQL Server database versions.

Oracle GoldenGate for SQL Server supports the mapping, filtering, and transformation of source data, unless noted otherwise in this document, as well as replicating data derived from other source databases supported by Oracle GoldenGate, into SQL Server databases.

This part describes tasks for configuring and running Oracle GoldenGate for SQL Server.

Topics:

26

Understanding What's Supported for SQL Server

This chapter contains information on database and table features supported by Oracle GoldenGate for SQL Server.

Topics:

Instance Requirements

- The SQL Server server name (`@@SERVERNAME`) must not be `NULL`.
- (Extract) For Oracle GoldenGate to capture transactional data, the SQL Server Agent must be running on the source SQL Server instance and the SQL Server Change Data Capture job must be running against the database. If SQL Server Transactional Replication is also enabled for the database, then the SQL Server Log Reader Agent must be running.
- If your data for `TEXT, NTEXT, IMAGE,` or `VARCHAR(MAX), NVARCHAR(MAX)` and `VARBINARY(MAX)` columns will exceed the SQL Server default size set for the `max text repl size` option, then extend the size. Use `sp_configure` to view or adjust the current value of `max text repl size`.

  > **Note:**
  >
  > For Amazon RDS for SQL Server, to adjust instance settings, you need to use Parameter Groups instead of `sp_configure`.

- It is recommended to install the most recent Service Pack or Cummulative Update for your SQL Server instance to ensure proper functionality. For SQL Server 2012, 2014, 2016, and 2017, Microsoft has identified and fixed several important issues that directly affect the SQL Server Change Data Capture feature. This situation impacts the ability for Oracle GoldenGate to correctly capture data. The current known issues that require Microsoft patches include KB3030352, KB3166120, and KB4073684.

Database Requirements

Observe the following requirements and limitations for supporting Oracle GoldenGate:

- Only user databases are supported for capture and delivery.
- Ensure that `Auto Create Statistics` and `Auto Update Statistics` are enabled for the database.
- The database must be set to the compatibility level of the SQL Server instance version.
- Oracle GoldenGate supports SQL Server databases configured with Transparent Data Encryption (TDE).
- (Extract) The source database can be set to any recovery model that supports the change data capture feature in Microsoft SQL Server.
- If the source database was created by restoring a backup from a different instance you must synchronize the database owner SID with the SID on the new instance. Alternatively, you can use `sp_changedbowner` to set the restored database to a current login.
- (AlwaysOn) Extract supports capturing from the primary database, or a read-only, synchronous-commit mode. Asynchronous-commit mode are not supported for capture.
- Replicat performance consideration: Beginning with SQL Server 2016, Microsoft changed the default setting for the database option `TARGET_RECOVERY_TIME` from 0 to 60 seconds. It has been demonstrated in internal testing that this can reduce the Replicat's throughput. If you experience Replicat throughput degradation, consider adjusting the `TARGET_RECOVERY_TIME` setting to 0.

Limitations:

- Oracle GoldenGate does not support capture or delivery of system databases.
- Oracle GoldenGate does not support capture from contained databases.
- Source database names cannot exceed 121 characters. This limitation is due to the SQL Server stored procedures that are used to enable supplemental logging.
- If you are configuring the Oracle GoldenGate heartbeat functionality, the SQL Server database name must not exceed 107 characters.
- Capture from SQL Server databases enabled with In-Memory OLTP (in-memory optimization) is not supported. When you add a Memory Optimized Data file group to your database, Oracle GoldenGate is not allowed to enable supplemental logging for any table in the database. Conversely, if supplemental logging has been enabled for any table in the database prior to the creation of a Memory Optimized Data file group, SQL Server does not allow a Memory Optimized Data file group to be created.
- (AlwaysOn) Capture from databases configured in asynchronous-commit mode of an AlwaysOn Availability group are not supported.

Table Requirements

Tables to be included for capture and delivery must include only the data types that are listed in Supported SQL Server Data Types.

- Oracle GoldenGate supports capture of transactional DML from user tables, and delivery to user tables and writeable views.
- DDL operations are not supported.
- Oracle GoldenGate supports the maximum permitted table names and column lengths for tables that are tracked by SQL Server Change Data Capture.
- The sum of all column lengths for a table to be captured from must not exceed the length that SQL Server allows for enabling Change Data Capture for the table. If the sum of all column lengths exceeds what is allowed by SQL Server procedure `sys.sp.cdc_enable_table`, then `ADD TRANDATA` cannot be enabled for that table. The maximum allowable record length decreases as more columns are present, so there is an inverse relationship between maximum record length and the number of columns in the table.

Supported SQL Server Data Types

The following data types are supported for capture and delivery, unless specifically noted in the limitations that follow:

- Binary Data Types
  - `(binary, varbinary, varbinary (max))`
  - `(varbinary (max)` with `FILESTREAM)`
- Character Data Types
  - `(char, nchar, nvarchar, nvarchar (max), varchar, varchar (max))`
- Date and Time Data Types
  - `(date, datetime2, datetime, datetimeoffset, smalldatetime, time)`
- Numeric Data Types
  - `(bigint, bit, decimal, float, int, money, numeric, real, smallint, smallmoney, tinyint)`
- LOBs
  - `(image, ntext, text)`
- Other Data Types
  - `(timestamp, uniqueidentifier, hierarchyid, geography, geometry, sql_variant (Delivery only), XML)`
- Oracle GoldenGate for SQL Server can replicate column data that contains `SPARSE` settings..

Limitations:

- Oracle GoldenGate does not support filtering, column mapping, or manipulating large objects larger than 4KB. Full Oracle GoldenGate functionality can be used for objects of up to 4KB.
- Oracle GoldenGate treats XML data as a large object (LOB), as does SQL Server when the XML does not fit into a row. SQL Server extended XML enhancements (such as lax validation, `DATETIME` , union functionality) are not supported.
- A system-assigned `TIMESTAMP` column or a non-materialized computed column cannot be part of a key. A table containing a `TIMESTAMP` column must have a key, which can be a primary key or unique constraint, or a substitute key specified with a `KEYCOLS` clause in the `TABLE` or `MAP` statements. For more information see Assigning Row Identifiers.
- Oracle GoldenGate supports multibyte character data types and multi byte data stored in character columns. Multibyte data is supported only in a like-to-like, SQL Server configuration. Transformation, filtering, and other types of manipulation are not supported for multibyte character data.
- If capture of data for `TEXT`, `NTEXT`, `IMAGE`, `VARCHAR (MAX)`, `NVARCHAR(MAX)` and `VARBINARY (MAX)` columns will exceed the SQL Server default size set for the `max text repl size` option, extend the size. Use `sp_configure` to view the current value of `max text repl size` and adjust the option as needed.

  > **Note:**
  >
  > Amazon RDS for SQL Server does not allow `max text repl size` to be greater than 64MB.

- Columns of `IMAGE`, `NTEXT`, and `TEXT` data types are logged as a `NULL` value for delete and before image update operations. Columns of `VARBINARY(MAX)`, `VARCHAR(MAX)`, and `NVARCHAR(MAX)` are logged as a `NULL` value for before image update operations unless the column was updated.

  For more information, review the Large Object Data Types content in the following Microsoft document:

  https://docs.microsoft.com/en-us/sql/relational-databases/system-tables/cdc-capture-instance-ct-transact-sql?view=sql-server-ver15
- Oracle GoldenGate supports UDT and UDA data of up to 2 GB in size. All UDTs except SQL_Variant are supported.
- Common Language Runtime (CLR), including SQL Server built-in CLR data types (such as, geometry, geography, and hierarchy ID), are supported. CLR data types are supported only in a like-to-like SQL Server configuration. Transformation, filtering, and other types of manipulation are not supported for CLR data.
- `VARBINARY (MAX)` columns with the `FILESTREAM` attribute are supported up to a size of 4 GB. Extract uses standard `Win32` file functions to read the `FILESTREAM` file.
- The range and precision of floating-point numbers depends on the host machine. In general, precision is accurate to 16 significant digits, but you should review the

database documentation to determine the expected approximations. Oracle GoldenGate rounds or truncates values that exceed the supported precision.

- Oracle GoldenGate supports time stamp data from 0001/01/03:00:00:00 to 9999/12/31:23:59:59. If a time stamp is converted from GMT to local time, these limits also apply to the resulting time stamp. Depending on the time zone, conversion may add or subtract hours, which can cause the time stamp to exceed the lower or upper supported limit.

Limitations on Computed Columns:

- Computed columns, either persisted or non-persisted, are not supported by Microsoft's Change Data Capture. Therefore, no data is written to the trail for columns that contain computed columns. To replicate data for non-persisted computed columns, use the `FETCHCOLS` or `FETCHMODCOLS` option of the `TABLE` parameter to fetch the column data from the table. Keep in mind that there can be discrepancies caused by differences in data values between the time that the column was changed in the data base and the time that Extract fetches the data for the transaction record that is being processed.
- Replicat does not apply DML to any computed column, even if the data for that column is in the trail, because the database does not permit DML on that type of column. Data from a source persisted computed column, or from a fetched non- persisted column, can be applied to a target column that is not a computed column.
- In an initial load, all of the data is selected directly from the source tables, not the transaction log. Therefore, in an initial load, data values for all columns, including non-persisted computed columns, is written to the trail or sent to the target, depending on the method that is used. As when applying change data, however, Replicat does not apply initial load data to computed columns, because the database does not permit DML on that type of column.
- Oracle GoldenGate does not permit a non-persisted computed column to be used in a `KEYCOLS` clause in a `TABLE` or `MAP` statement.
- If a unique key includes a non-persisted computed column and Oracle GoldenGate must use the key, the non-persisted computed column is ignored. This may affect data integrity if the remaining columns do not enforce uniqueness.
- If a unique index is defined on any non-persisted computed columns, it is not used.
- If a unique key or index contains a non-persisted computed column and is the only unique identifier in a table, Oracle GoldenGate must use all of the columns as an identifier to find target rows. Because a non-persisted computed column cannot be used in this identifier, Replicat may apply operations containing this identifier to the wrong target rows.

Non-Supported SQL Server Data Types and Features

- `SQL_Variant` data type is not supported for capture.
- Tables that contain unsupported data types may cause Extract to Abend. As a workaround, you must remove `TRANDATA` from those tables and remove them from the Extract's `TABLE` statement, or use the Extract's `TABLEEXCLUDE` parameter for the table.

Supported Objects and Operations for SQL Server

The following objects and operations are supported:

- Parallel Replicat is supported with Oracle GoldenGate for SQL Server.
- Oracle GoldenGate supports capture of transactional DML from user tables and delivery to user tables and writeable views.
- `TEXT`, `NTEXT`, `IMAGE`, `VARBINARY`, `VARBINARY (MAX)` `VARCHAR (MAX)`, and `NVARCHAR (MAX)` columns are supported in their full size for operations that are logged by SQL Server Chang Data Capture. For example, columns of `IMAGE`, `NTEXT`, and `TEXT` data types are logged as a NULL value for delete operations. For more information, review the Large Object Data Types content at the following Microsoft document:

  https://docs.microsoft.com/en-us/sql/relational-databases/system-tables/cdc-capture-instance-ct-transact-sql?view=sql-server-ver15
- Oracle GoldenGate supports the maximum row sizes that are permitted for tables that are enabled for SQL Server Change Data Capture.
- Oracle GoldenGate supports capture from tables enabled with `PAGE` and `ROW` compression. For partitioned tables that use compression, all partitions must be enabled with the same compression type.
- Oracle GoldenGate supports capture for partitioned tables if the table has the same physical layout across all partitions.
- The sum of all column lengths for a table to be captured from must not exceed the length that SQL Server allows for enabling Change Data Capture for the table. If the sum of all column lengths exceeds what is allowed by the SQL Server procedure `sys.sp.cdc_enable_table`, then `ADD TRANDATA` cannot be added for that table. The maximum allowable record length decreases as more columns are present, so there is an inverse relationship between maximum record length and the number of columns in the table.

Non-Supported Objects and Operations for SQL Server

The following objects and operations are not supported:

- For source databases, operations that are not supported by SQL Server Change Data Capture, such as `TRUNCATE` statements. Refer to Microsoft SQL Server Documentation for a complete list of the operations that are limited by enabling SQL Server Change Data Capture.
- Oracle GoldenGate for SQL Server does not support the capture or delivery of DDL changes for SQL Server and extra steps are required for Oracle GoldenGate processes on the source and target to handle any table level DDL changes, including table index rebuild operations. See Requirements for Table Level DDL Changes.
- Views are not supported.
- Operations by the `TextCopy` utility and `WRITETEXT` and `UPDATETEXT` statements. These features perform operations that either are not logged by the database or are only partially logged, so they cannot be supported by the Extract process.
- Partitioned tables that have more than one physical layout across partitions.
- Partition switches against a source table. SQL Server Change Data Capture treats partition switches as DDL operations, and the data moved from one partition to another is not logged in the CDC tables, so you must follow the procedures in

Requirements for Table Level DDL Changes to manually implement a partition switch when the table is enabled for supplemental logging.

- Due to a limitation with SQL Server's Change Data Capture, column level collations that are different from the database collation, may cause incorrect data to be written to the CDC tables for character data and Extract will capture them as they are written to the CDC tables. It is recommended that you use `NVARCHAR`, `NCHAR` or `NTEXT` data type for columns containing non-ASCII data or use the same collation for table columns as the database. For more information see, About Change Data Capture (SQL Server).
- Due to a limitation with SQL Server's Change Data Capture, `NOOPUPDATES` are not captured by the SQL Server Change Data Capture agent so there are no records for Extract to capture for no-op update operations.
- Temporal tables are not supported for enabling Change Data Capture, therefore cannot be configured for Extract for source implementations.
- Requirements for Table Level DDL Changes

Requirements for Table Level DDL Changes

Oracle GoldenGate for SQL Server does not support the capture or delivery of DDL changes. However, beginning with Oracle GoldenGate 21c, changes made to tables enabled with `TRANDATA` will not cause Extract to abend. Extract will continue to process change data for the table as it existed when `TRANDATA` was enabled.

Operations considered to be table-level DDL changes include, but are not limited to: `ALTER TABLE`, `TRUNCATE TABLE`, index rebuilds, and partition switches.

To avoid data inconsistencies due to table level DDL changes, the following steps are required.

1. Source: Pause or Stop application data to the table or tables to be modified.
2. Source: Ensure that there are no open transactions against the table to be modified.
3. Source: Ensure that the SQL Server CDC Capture job processes all remaining transactions for the table that is to be modified.
4. Source: Ensure that the Extract processes all the transactions for the table that is to be modified, prior to making any DDL changes.
5. Target: Ensure that the Replicat processes all the transactions for the table that is to be modified, prior to making any DDL changes.
6. Optionally, implementing an Event Marker table can be used to determine when all of the remaining transactions have been processed for the table that is to be modified, and handle the coordination of when to correctly stop the Extract and Replicat.
7. Source: Stop the Extract process.
8. Target: Stop the Replicat process.
9. Source: Disable supplemental logging for the table to be modified by running `DELETE TRANDATA`.
10. Source: Make table DDL changes to the source table.
11. Target: Make table DDL changes to the target table.
12. Source: Re-enable supplemental logging by running `ADD TRANDATA` to the table(s) after the modifications have been performed.

13. Source: Start the Extract.
14. Target: Start the Replicat.
15. Source: Resume application data to the table or tables that were modified.

System Schemas for SQL Server

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

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

27

Preparing the System for Oracle GoldenGate

This chapter contains steps to take so that the database with which Oracle GoldenGate interacts is correctly configured to support Oracle GoldenGate capture and delivery. Some steps apply only to a source system, some only to a target, and some to both.
Topics:

Prepare Database Users and Privileges

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

Oracle GoldenGate for SQL Server

Oracle GoldenGate processes require a database user in order to capture from and apply data to a SQL Server database and it is recommended to create a dedicated database user to be used exclusively by Oracle GoldenGate processes.

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

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

SQL Server and Azure SQL Managed Instance

The following user requirements and minimum database privileges and permissions are required for Oracle GoldenGate to capture from and apply to a SQL Server or Azure SQL Managed Instance database.

1. Create a dedicated login for Oracle GoldenGate for SQL Server or Azure SQL Managed Instance.
2. Add the login as a user to the `msdb` database and to the source or target database.
3. Create a schema in the source or target database, to be used for objects required for Oracle GoldenGate. This schema should map to the `GGSCHEMA` value used in the `GLOBALS` parameter file.
4. Enable the following privileges and permissions for the Oracle GoldenGate user based on whether the user is for an Extract, or for a Replicat.

Table 27-1 Privileges and Permissions for Oracle GoldenGate User

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

`SYSADMIN`	Yes	No	Required for a one time change to enable database level Change Data Capture (CDC) if not already enabled and can be revoked once `TRANDATA` has been enabled.
			`ALTER SERVER ROLE sysadmin ADD MEMBER gguser;`
			Database Administrators with sysadmin credentials can manually enable the database for CDC using the following, which would negate the need for the Extract user to have this privilege:
			`EXEC msdb.sys.sp_cdc_enable_db 'source_database'`

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

Amazon RDS User Permissions and Requirements

The following user requirements and minimum database privileges and permissions are required for Oracle GoldenGate to capture from and apply to an Amazon RDS for SQL Server database:

1. Create a dedicated login for Oracle GoldenGate for Amazon RDS for SQL Server.

2. Add the login as a user to the `msdb` database and to the source or target database.
3. Create a schema in the source or target database, to be used for objects required for Oracle GoldenGate. This schema should map to the `GGSCHEMA` value used in the `GLOBALS` parameter file.
4. Enable the following privileges and permissions for the Oracle GoldenGate user based on whether the user is for an Extract, or for a Replicat.

Table 27-2 Privileges and Permissions for Oracle GoldenGate User

Privilege	Extract	Replicat	Syntax
msdb Database Roles and Privileges			

`EXECUTE ON rds_cdc_enable_db`	Yes	No	`GRANT EXECUTE ON msdb.dbo.rds_cdc_enable_db TO gguser;`
			Database administrators with master credentials can manually enable the database for Change Data Capture using the following command, which would negate the need for the Extract user to have this permission:
			`EXEC msdb.dbo.rds_cdc_enable_db 'source_database'`

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

Azure SQL Database

The following user requirements and minimum database privileges and permissions are required for Oracle GoldenGate to apply to an Azure SQL Database:

1. Create a dedicated login for Oracle GoldenGate for Azure SQL Database.
2. Add the login as a user to the target database.
3. Create a schema in the target database, to be used for objects required for Oracle GoldenGate. This schema should map to the `GGSCHEMA` value used in the `GLOBALS` parameter file.
4. Enable the following privileges and permissions for the Oracle GoldenGate user.

Table 27-3 Privileges and Permissions for Oracle GoldenGate User

Privilege	Extract	Replicat	Syntax
User Database Roles and Privileges			
DBOWNER	NA	Yes	`ALTER ROLE db_owner ADD MEMBER gguser;`

Google Cloud SQL for SQL Server

The following user requirements and minimum database privileges and permissions are required for Oracle GoldenGate to capture from and apply to a Google Cloud SQL for SQL Server database:

1. Create a dedicated login for Oracle GoldenGate Google Cloud SQL for SQL Server. The user must be created from within the **Users** section of the Google Cloud dashboard for the database instance.
2. Add the user to the source or target database.
3. Create a schema in the source or target database, to be used for objects required for Oracle GoldenGate. This schema should map to the `GGSCHEMA` value used in the `GLOBALS` parameter file.
4. If the database is to be used as a source for an Extract, manually enable the database for Change Data Capture (CDC):

   ```
   EXEC msdb.dbo.gcloudsql_cdc_enable_db 'source_database';
   ```

5. Enable the following privileges and permissions for the Oracle GoldenGate user based on whether the user is for an Extract, or for a Replicat.

Table 27-4 Privileges and Permissions for Oracle GoldenGate User

Privilege	Extract	Replicat	Syntax
User Database Roles and Privileges			
DBOWNER	Yes	Yes	`ALTER ROLE db_owner ADD MEMBER gguser;`

Configure a Database Connection

This section contains instructions for setting up the Extract and Replicat connections to a DB2 z/OS database.

Topics:

Extract and Replicat Database Connectivity

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

Creating a Database Connection on Linux

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

Select the following link for download and installation steps:

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

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

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

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

   ```
   vi odbc_template_file.ini
   ```

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

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

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

3. Install the data source using the following command.

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

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

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

Creating a Database Connection on Windows

Before creating a database connection for Oracle GoldenGate processes running on Windows, install the latest version of Microsoft ODBC Driver for SQL Server.

Follow these steps to create a system DSN on the Windows server where Oracle GoldenGate is installed.

To create a SQL Server DSN

1. Open the ODBC Data Sources (64-bit) application.
2. In the ODBC Data Source Administrator dialog box, select the **System DSN** tab, and then click **Add**.
3. Under Create New Data Source, select the **ODBC Driver {*version*} for SQL Server** and then click **Finish**. The **Create a New Data Source to SQL Server** wizard appears.
4. Enter the following details, and click **Next**:
   - Name: Can be of your choosing. In a Windows cluster, use one name across all nodes in the cluster.
   - Description: (Optional) Type a description of this data source.
   - Server: Type the SQL Server connection string or server\instance name. For Always On connections, use the **listener\instance name** of the Always On Availability Group.
5. For login authentication, select one of the following options, and then click **Next**:
   a. With Integrated Windows Authentication
   b. With SQL Server authentication using a login ID and password entered by the user
6. Select **Change the default database to**, and then select the source or target database from the list. Enable the **Use ANSI** settings. Click **Next**.
7. Leave the next page set to the defaults. Click **Finish**.
8. Click **Test Data Source** to test the connection.
9. If the test is successful, close the confirmation box and the **Create a New Data Source** box.
10. Repeat this procedure for each SQL Server source and target database.

Connecting to the Listener of a SQL Server Always On Configuration

Extract and Replicat can connect to the listener of an Always On configuration or directly to the current primary replica of the group, depending on the DSN connection used.

The advantage of creating the connection to the listener is that Extract or Replicat can reconnect to the new primary replica upon failover without having to reconfigure the DSN to the new primary.

An Extract can also be configured to route its read-only queries to an available readable, synchronous mode secondary replica. By default, if Extract connects to a listener, all processing will be done against the primary replica, but if an Extract is configured with the `TRANLOGOPTIONS ALWAYSONREADONLYROUTING` parameter, its read-only queries are routed by the listener to an available readable secondary replica.

See `TRANLOGOPTIONS` and Requirements Summary for Capture and Delivery of Databases in an AlwaysOn Availability Group for more information.

If creating the DSN to connect to a Listener of an Always On configuration, enable the Multi-subnet failover option when creating the DSN. For Linux DSN connections, use the `MultiSubnetFailover=Yes` option in the DSN entry.

Preparing Tables for Processing

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

- Disabling Triggers and Cascade Constraints on the Target
- Replicat Consideration for Target Identity Columns, Triggers, and Constraints
- Setting the NOT FOR REPLICATION flag for Target Identity Columns, Triggers, and Constraints
- Ensuring Row Uniqueness in Source and Target Table
- Improving IDENTITY Replication with Array Processing

Disabling Triggers and Cascade Constraints on the Target

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

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

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

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

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

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

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

Replicat Consideration for Target Identity Columns, Triggers, and Constraints

When replicating data to a target SQL Server database that has identity columns, triggers, and cascade and check constraints, consider the following:

- For columns that have an identity column, Replicat sets the `IDENTITY_INSERT ON` for the table, which may reduce delivery performance.
- For tables that have triggers or cascade constraints, execution of the trigger or cascade operation may result in a Replicat error if the Replicat is configured to deliver the same data that a trigger will insert or cascade constraint will update or delete.

  For example, `TableA` on the source has a trigger that inserts a record into `TableB`. The Extract is configured to capture records for both TableA and TableB. On the target, the Replicat will first insert a record for `TableA`, then the trigger for `TableA` fires and inserts into `TableB`, followed by the Replicat attempting to insert the same record into `TableB`, which will result in a Replicat error.
- Check any foreign key constraints are also enforced, which may reduce delivery performance.

To overcome these situations, there are several options that can be implemented based on the replication use case.

- For unidirectional implementations where a Replicat is the only process writing data to the target tables, consider the following options for Identity columns, triggers and constraints on the target tables.
  1. Disable or drop the Identity property, triggers and constraints on the target tables.
  2. Modify the identity property, triggers and constraints and set the `NOT FOR REPLICATION` option on for each and ensure that the Microsoft ODBC driver is at least version 17.8.1.
- For multi-directional implementations where both a Replicat and application write data to the target tables, and triggers and constraints are enabled, modify the Identity property, triggers and constraints and set the `NOT FOR REPLICATION` option on for each and ensure that the Microsoft ODBC driver is at least version 17.8.1.

  Additionally, to use `IDENTITY` columns in a multi-directional replication configuration, define the `IDENTITY` columns to have an increment value equal to the number of servers in the configuration, with a different seed value for each one.

  For example, a three-database configuration would be as follows:

  Database1 set the seed value at 0 with an increment of 3.

  Database2 set the seed value at 1 with an increment of 3.

  Database3 set the seed value at 2 with an increment of 3.

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

1. Set the `NOT FOR REPLICATION` flag on the following objects.
   - Foreign key constraints
   - Check constraints
   - `IDENTITY` columns

- Triggers (requires textual changes to the definition. See the SQL Server documentation for more information.)

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

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

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

Ensuring Row Uniqueness in Source and Target Table

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

Unless a `KEYCOLS` clause is used in the `TABLE` or `MAP` statement, Oracle GoldenGate selects a row identifier to use in the following order of priority:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. If none of the preceding key types exist (even though there might be other types of keys defined on the table) Oracle GoldenGate constructs a pseudo key of all columns that the database allows to be used in a unique key, excluding those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration.

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

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

- Using KEYCOLS to Specify a Custom Key

Using KEYCOLS to Specify a Custom Key

If a table does not have an applicable row identifier, or if you prefer that identifiers are not used, you can define a substitute key, providing that the table has columns that always contain unique values. You define this substitute key by including a KEYCOLS clause within the Extract TABLE parameter and the Replicat MAP parameter. The specified key overrides any existing primary or unique key that Oracle GoldenGate finds.

Improving IDENTITY Replication with Array Processing

Because only one table per session can have IDENTITY_INSERT set to ON, Replicat must continuously toggle IDENTITY_INSERT when it applies IDENTITY data to multiple tables in a session. To improve the performance of Replicat in this situation, use the BATCHSQL parameter. BATCHSQL causes Replicat to use array processing instead of applying SQL statements one at a time.

Globalization Support

Oracle GoldenGate provides globalization support that lets it process data in its native language encoding. The Oracle GoldenGate apply process (Replicat) can convert data from one character set to another when the data is contained in character column types.

28

Preparing the Database for Oracle GoldenGate — CDC Capture

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

You can learn more about CDC Capture with this Oracle By Example:

Using the Oracle GoldenGate for SQL Server CDC Capture Replication http://www.oracle.com/webfolder/technetwork/tutorials/obe/fmw/goldengate/12c/sql_cdcrep/sql_cdcrep.html.

Topics:

Enabling CDC Supplemental Logging

With the CDC Extract, the method of capturing change data is via SQL Server Change Data Capture tables, so it is imperative that you follow the procedures and requirements below, so that change data is correctly logged, maintained, and captured by Extract.

You will enable supplemental logging with the `ADD TRANDATA` command so that Extract can capture the information that is required to reconstruct transactions.

`ADD TRANDATA` must be issued for all tables that are to be captured by Oracle GoldenGate, and to do so requires that a valid schema be used in order to create the necessary Oracle GoldenGate tables and stored procedures.

Enabling supplemental logging for a CDC Extract does the following:

- Enables SQL Server Change Data Capture at the database level, if it's not already enabled.
  - `EXECUTE sys.sp_cdc_enable_db`
- Creates a Change Data Capture staging table for each base table enabled with supplemental logging by running `EXECUTE sys.sp_cdc_enable_table`, and creates a trigger for each CDC table. The CDC table exists as part of the system tables within the database and has a naming convention like, `cdc.OracleGG_basetableobjectid_CT`.
- Creates a tracking table of naming convention *`schema`*`.OracleGGTranTables`. This table is used to store transaction indicators for the CDC tables, and is populated when the trigger for a CDC table is fired. The table will be owned by the schema listed in the `GLOBALS` file's, `GGSCHEMA` parameter.
- Creates a unique fetch stored procedure for each CDC table, as well as several other stored procedures that are required for Extract to function. These stored procedures will be owned by the schema listed in the `GLOBALS` file's, `GGSCHEMA` parameter.

- Also, as part of enabling CDC for tables, SQL Server creates two jobs per database:

  `cdc.dbname_capture`

  `cdc.dbname_cleanup`

- The CDC Capture job is the job that reads the SQL Server transaction log and populates the data into the CDC tables, and it is from those CDC tables that the Extract will capture the transactions. So it is of extreme importance that the CDC capture job be running at all times. This too requires that SQL Server Agent be set to run at all times and enabled to run automatically when SQL Server starts.
- Important tuning information of the CDC Capture job can be found in CDC Capture Method Operational Considerations.
- The CDC Cleanup job that is created by Microsoft does not have any dependencies on whether the Oracle GoldenGate Extract has captured data in the CDC tables or not. Therefore, extra steps need to be followed in order to disable or delete the CDC cleanup job immediately after `TRANDATA` is enabled, and to enable Oracle GoldenGate's own CDC cleanup job. See Retaining the CDC Table History Data for more information.

The following steps require a database user who is a member of the SQL Server System Administrators (`sysadmin`) role.

1. In the source Oracle GoldenGate installation, ensure that a `GLOBALS` (all CAPS and no extension) file has been created with the parameter `GGSCHEMA schemaname`. Ensure that the schema name used has been created (`CREATE SCHEMA schemaname`) in the source database. This schema will be used by all subsequent Oracle GoldenGate components created in the database, therefore it is recommended to create a unique schema that is solely used by Oracle GoldenGate, such as `'ogg'`. It is recommended not to use the SQL Server schema **cdc** and to create a new schema specific to Oracle GoldenGate.
2. On the source system, run `GGSCI`
3. Issue the following command to log into the database:

   `DBLOGIN SOURCEDB DSN [,{USERID user, PASSWORD password | USERIDALIAS alias}]`
   Where:

   - `SOURCEDB DSN` is the name of the SQL Server data source.
   - `USERID user` is the database login and `PASSWORD password` is the password that is required if the data source connects via SQL Server authentication. Alternatively, `USERIDALIAS alias` is the alias for the credentials if they are stored in a credentials store. If using `DBLOGIN` with a DSN that is using Integrated Windows authentication, the connection to the database for the GGSCI session will be that of the user running GGSCI. In order to issue `ADD TRANDATA` or `DELETE TRANDATA`, this user must be a member of the SQL Server `sysadmin` server role.
4. In GGSCI, issue the following command for each table that is, or will be, in the Extract configuration. You can use a wildcard to specify multiple table names.

   `ADD TRANDATA owner.table`

   `ADD TRANDATA owner.*`

Optionally, you can designate the filegroup in which the SQL Server Change Data Capture staging tables will be placed, by using the `FILEGROUP` option with an existing filegroup name.

```
ADD TRANDATA owner.table FILEGROUP cdctables
```

See `ADD TRANDATA`

Purging CDC Staging Data

When enabling supplemental logging, data that is required by Extract to reconstruct transactions are stored in a series of SQL Server CDC system tables, as well Oracle GoldenGate objects that are used to track operation ordering within a transaction. These tables require routine purging in order to reduce data storage within the database. As part of enabling supplemental logging using `TRANDATA`, SQL Server creates its own Change Data Capture Cleanup job, however this job is unaware that an Extract may still require data from these CDC system tables and can remove that data before the Extract has a chance to capture it.

If data that Extract needs during processing has been deleted from the CDC system tables, then one of the following corrective actions might be required:

- Alter Extract to capture from a later point in time for which CDC data is available (and accept possible data loss on the target).
- Resynchronize the source and target tables, and then start the Oracle GoldenGate environment over again.

To remedy the situation, Oracle GoldenGate for SQL Server requires a set of stored procedures and tables that are used to purge the old CDC staging data while ensuring that no data is purged that an Extract has yet to process.

Use the following steps immediately after enabling supplemental logging and prior to starting the Extract, to create the Oracle GoldenGate CDC Cleanup job and associated objects. You can re-run these steps to enable this feature should any of the objects get manually deleted.

To create the Oracle GoldenGate CDC Cleanup job and objects:

The `ogg_cdc_cleanup_setup` file is located in the Oracle GoldenGate home directory.

The script uses the Microsoft `sqlcmd` utility, so ensure that `sqlcmd` is installed on the system where Oracle GoldenGate is installed.

The script also requires a SQL Server authenticated database user who is a member of the SQL Server System Administrators (`sysadmin`) role. Windows authentication is not supported for the script.

1. Manually stop and disable the database's SQL Server `cdc.`*`dbname`*`_cleanup` job from SQL Server Agent. Alternatively, you can drop it from the source database with the following command.

   ```
   EXECUTE sys.sp_cdc_drop_job 'cleanup'
   ```

2. Run the `ogg_cdc_cleanup_setup` file, providing the following variable values.

For Windows:

```
ogg_cdc_cleanup_setup.bat createJob userid password databasename
servername\instancename schema
```

For Linux:

```
./ogg_cdc_cleanup_setup.sh createJob userid password databasename
"servername,port" schema
```

In the preceding examples, USER ID and password should be a valid SQL Server login and password for a user, which has `sysadmin` rights. The `databasename`, `servername\instancename`, or `servername`, `port`, are the source database name, server, and instance, or server and TCP/IP port where SQL Server is running. If only the server name is listed, then the default instance will be connected to. The schema is the schema name listed in the GLOBALS file, with the `GGSCHEMA` parameter. This schema should be the same for all Oracle GoldenGate objects, including supplemental logging, checkpoint tables, heartbeat tables, and the Oracle GoldenGate CDC Cleanup job.

For example:

```
ogg_cdc_cleanup_setup.bat createJob ggsuser ggspword db1
server1\inst1 ogg
```

When using `server,port` in the connection string, enclose the string in double quotes, for example:

```
ogg_cdc_cleanup_setup.bat createJob login password source database
"sql2016.samplestring.us-west-1.rds.amazonaws.com,1433" OGG schema
name
```

The Oracle GoldenGate CDC Cleanup job when created, is scheduled to run every ten minutes, with a default retention period of either seventy two hours or one hour, depending on the patch version of Oracle GoldenGate. The job will not purge data for an Extract's recovery checkpoint however, regardless of the retention period.

Additional information of the Oracle GoldenGate CDC Cleanup job can be found in CDC Capture Method Operational Considerations.

Enabling Bi-Directional Loop Detection

Loop detection is a requirement for bi-directional implementations of Oracle GoldenGate, so that an Extract for one source database does not recapture transactions sent by a Replicat from another source database.

With the CDC Extract capture method, by default, any transaction committed by a Replicat into a database where an Extract is configured, will recapture that transaction from the Replicat as long as supplemental logging is enabled for those tables that the Replicat is delivering to.

In order to ignore recapturing transactions that are applied by a Replicat, you must use the `TRANLOGOPTIONS FILTERTABLE` parameter for the CDC Extract. The table used as

the filtering table will be the Oracle GoldenGate checkpoint table that you must create for the Replicat.

To create a Filter Table and enable Supplemental Logging:

The steps below require a database user who is a member of the SQL Server System Administrators (`sysadmin`) role.

1. On the source system, run `GGSCI`
2. Issue the following command to log into the database.

   ```
   DBLOGIN SOURCEDB DSN [,{USERID user, PASSWORD password | USERIDALIAS alias}]
   ```

   In the preceding example, the `SOURCEDB DSN` is the name of the SQL Server data source. The `USERID` user is the database login and `PASSWORD` password is the password that is required if the data source connects through SQL Server authentication. Alternatively, `USERIDALIAS` alias is the alias for the credentials if they are stored in a credentials store. If using `DBLOGIN` with a DSN that is using Integrated Windows authentication, the connection to the database for the GGSCI session is that of the user running GGSCI. In order to issue `ADD TRANDATA` or `DELETE TRANDATA`, this user must be a member of the SQL Server `sysadmin` server role.
3. Create the Oracle GoldenGate checkpoint table that is used by the Replicat to deliver data to the source database.

   Example: `ADD CHECKPOINTTABLE ogg.ggchkpt`

   It is recommended that you use the same schema name as used in the `GGSCHEMA` parameter of the `GLOBALS` file.
4. Enable supplemental logging for the newly created checkpoint table.

   Example: `ADD TRANDATA ogg.ggchkpt`
5. Add the Replicat with the checkpoint table information.

   Example: `ADD REPLICAT reptgt1, EXTTRAIL ./dirdat/e2,checkpointtable ogg.ggchkpt`
6. Configure the Extract with the `IGNOREREPLICATES` (on by default) and `FILTERTABLE` parameters, using the Replicat's checkpoint table for the filtering table.

   ```
   TRANLOGOPTIONS IGNOREREPLICATES
   ```

   ```
   TRANLOGOPTIONS FILTERTABLE ogg.ggchkpt
   ```

29

CDC Capture Method Operational Considerations

This section provides information about the SQL Server CDC Capture options, features, and recommended settings.

Topics:

Tuning SQL Server Change Data Capture

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

- Ensure that Auto Create Statistics and Auto Update Statistics are enabled for the database. Maintaining statistics on the `cdc.OracleGG_#####_CT` , `cdc.lsn_time_mapping`, and `OracleGGTranTables` table is crucial to the performance and latency of the Extract.
- The SQL Server Change Data Capture job collects data from the SQL Server transaction log and loads it into the Change Data Capture staging tables within the database.

  As part of the job that is created, there are several available tuning parameters that can be used, and information on how to best tune the job can be found in the following article: https://technet.microsoft.com/en-us/library/dd266396(v=sql.100).aspx

  As a general recommendation, you should change the SQL Server Change Data Capture Job polling interval from the default of 5 seconds to 1 second.

  To change the default polling interval of the CDC Capture job, execute the following queries against the database:

```
EXEC [sys].[sp_cdc_change_job]
@job_type = N'capture',
@pollinginterval = 1,
GO,
--stops cdc job
EXEC [sys].[sp_cdc_stop_job],
@job_type = N'capture',
GO,
--restarts cdc job for new polling interval to take affect
EXEC [sys].[sp_cdc_start_job],
@job_type = N'capture',
```

Oracle GoldenGate CDC Object Versioning

Oracle GoldenGate provides a version tracking subsystem to track the CDC objects that are created by Oracle GoldenGate when enabling supplemental logging. These objects are:

- Oracle GoldenGate change tracking tables in the format `OracleGG_object id_CT`.
- Stored procedures in the format *`fetch_database name_object id`*
- Stored procedures `OracleCDCExtract`, `OracleGGCreateProcs`, and `OracleGGCreateNextBatch`.
- After successfully completing the `ADD TRANDATA` command, GGSCI creates a table called `OracleGGVersion` under the `GGSCHEMA` specified in the GLOBALS file, if it does not already exist.

  Next, GGSCI inserts a record into the table that tracks the start and end time of the `TRANDATA` session. When Extract starts up, it checks for consistency between itself and the Oracle GoldenGate CDC objects by comparing its internal version number with the version numbers found in the `OracleGGVersion` table. If it finds that the version numbers do not match, it abends with a message similar to the following:

  ```
  ERROR OGG-05337 The Oracle GoldenGate CDC object versions on database,
  source, are not consistent with the expected version, 2. The following
  versions(s) were found: 1. Rerun ADD TRANDATA for all tables
  previously enabled, including heartbeat, heartbeat seed, and filter
  tables.
  ```

Valid and Invalid Extract Parameters for SQL Server Change Data Capture

This section describes parameters used for the CDC Capture method. For more information about supported and unsupported parameters for the CDC Capture method, review *Reference for Oracle GoldenGate*.

`TRANLOGOPTIONS LOB_CHUNK_SIZE`

The Extract parameter `LOB_CHUNK_SIZE` is added for the CDC Capture method to support large objects. If you have huge LOB data sizes, then you can adjust the `LOB_CHUNK_SIZE` from the default of 4000 bytes, to a higher value up to 65535 bytes, so that the fetch size is increased, reducing the trips needed to fetch the entire LOB.

Example: `TRANLOGOPTIONS LOB_CHUNK_SIZE 8000`

`TRANLOGOPTIONS MANAGECDCCLEANUP/NOMANAGECDCCLEANUP`

The Extract parameter `MANAGECDCCLEANUP/NOMANAGECDCCLEANUP` is used by the CDC Capture method to instruct the Extract on whether or not to maintain recovery checkpoint data in the Oracle GoldenGate CDC Cleanup job. The default value is `MANAGECDCCLEANUP` and it doesn't have to be explicitly listed in the Extract. However, it

does require creating the Oracle GoldenGate CDC Cleanup job prior to starting the Extract. `MANAGECDCCLEANUP` should be used for all production environments, where `NOMANAGECDCCLEANUP` may be used for temporary and testing implementations as needed.

Example: `TRANLOGOPTIONS MANAGECDCCLEANUP`

`TRANLOGOPTIONS EXCLUDEUSER/EXCLUDETRANS`

The SQL Server CDC Capture job does not capture user information or transaction names associated with a transaction, and as this information is not logged in the CDC staging tables, Extract has no method of excluding DML from a specific user or DML of a specific transaction name. The `EXCLUDEUSER` and `EXCLUDETRANS` parameters are therefore not valid for the CDC Capture process.

`TRANLOGOPTIONS MANAGESECONDARYTRUNCATIONPOINT/NOMANAGESECONDARYTRUNCATIONPOINT/ACTIVESECONDARYTRUNCATIONPOINT`

The SQL Server Change Data Capture job is the only process that captures data from the transaction log when using the Oracle GoldenGate CDC Capture method. The secondary truncation point management is not handled by the Extract, and for the Change Data Capture Extract, these parameters are not valid.

`TRANLOGOPTIONS ALWAYSONREADONLYROUTING`

The `ALWAYSONREADONLYROUTING` parameter allows Extract for SQL Server to route its read-only processing to an available read-intent Secondary when connected to an Always On availability group listener.

`TRANLOGOPTIONS QUERYTIMEOUT`

Specifies how long queries to SQL Server will wait for results before reporting a timeout error message. This option takes an integer value to represent the number of seconds. The default query timeout value is 300 seconds (5 minutes). The minimum value is 0 seconds (infinite timeout). The maximum is 2147483645 seconds.

`TRANLOGOPTIONS TRANCOUNT`

Allows adjustment of the number of transactions processed per each call by Extract to pull data from the SQL Server change data capture staging tables. Based on your transaction workload, adjusting this value may improve capture rate throughput, although not all workloads will be positively impacted. The minimum value is 1, maximum is 100, and the default is 10.

Details of the Oracle GoldenGate CDC Cleanup Process

The Oracle GoldenGate CDC Cleanup job is required for a CDC Extract by default, since Extract defaults to `TRANLOGOPTIONS MANAGECDCCLEANUP`. It is installed from a Windows batch file (`ogg_cdc_cleanup_setup.bat`), which uses `sqlcmd` to connect to the source SQL Server database and create the necessary objects and job.

There should be one job for each database enabled for CDC Capture, and you must create the job and objects following the steps mentioned in the Preparing the Database for Oracle GoldenGate — CDC Capture section of this document.

Additional options for the utility are discussed in the following sections.

The steps below require a SQL Server authenticated database user who is a member of the SQL Server System Administrators (`sysadmin`) role. Windows authentication is not supported for the `.bat` batch file.

Removing an Extract from the Database

When the Oracle GoldenGate CDC Cleanup object tables exist, each CDC Extract that is started against that database will create an entry in the `OracleGGExtractCheckpoint` table. This entry tracks a particular Extract's point in time recovery checkpoint, which is used as the cutoff LSN for the Oracle GoldenGate CDC cleanup tasks. If there are multiple Extracts running, each logging more recent recovery checkpoints in the table, but one Extract has been removed from the system without removing its entry into the `OracleGGExtractCheckpoint` table, then no data will be purged newer than that deleted Extract's old recovery checkpoint for all of the CDC staging tables. So when deleting an Extract from the database, follow the steps below to remove the Extract from the `OracleGGExtractCheckpoint` table if more than one Extract is running against the database.

1. Log in to the Database with `DBLOGIN` from GGSCI:

   ```
   DBLOGIN SOURCEDB dsn_name USERIDALIAS alias_name
   ```

2. Stop the Extract:

   ```
   STOP EXTRACT extract_name
   ```

3. Delete the Extract:

   ```
   DELETE EXTRACT extract_name
   ```

By logging in to the database, the `DELETE EXTRACT` command removes the entry from the `OracleGGExtractCheckpoint` table, for that specific Extract.

Modifying the Oracle GoldenGate CDC Cleanup Job

The default schedule, retention period and operation batch size for the Oracle GoldenGate CDC Cleanup job of a database is to run every 10 minutes, with a data retention policy of 72 hours (listed as 4320 minutes), purging in batches of 500 records per transaction until the retention policy is meet, not to exceed the recovery checkpoint data of the Extract.

For variations in customer environments, or change data table data retention requirements, it may be necessary to adjust these properties to increase the purge batch size or to adjust retention policies and the job run-time schedule.

To adjust the job execution frequency, manually modify the schedule for the `OracleGGCleanup_dbname_Job` job within SQL Server Agent. If you need to adjust the retention period or purge batch size, you must manually edit the job step for the `OracleGGCleanup_dbname_Job` job within SQL Server Agent. The job step passes two parameters to the cleanup stored procedure, and you can modify the value for `@retention_minutes` to adjust the data retention policy as needed, or modify the `@threshold` value to increase or decrease the purge batch size. In high transactional environments, it may be necessary to increase the `@threshold` value to a number such as 10000. Monitoring the amount of time that it takes for the job to run within each cycle can be used to determine effective `@threshold` values.

Deleting the Oracle GoldenGate CDC Cleanup Job

If you no longer require the Oracle GoldenGate CDC Cleanup job and associated objects and need to remove them, perform the following steps:

1. Open a command prompt and change to the Oracle GoldenGate installation folder.

2. Run the `ogg_cdc_cleanup_setup.bat` file, providing the following variable values:

```
ogg_cdc_cleanup_setup.bat dropJob userid password databasename
servername\instancename schema
```

Example: `ogg_cdc_cleanup_setup.bat dropJob ggsuser ggspword db1 server1\inst1 ogg`

Changing from Classic Extract to a CDC Extract

If you plan to change from using a Classic Extract from Oracle GoldenGate 12c (12.3.0.1) or earlier, to an Oracle GoldenGate 21c CDC Extract, then you must remove the supplemental logging that was implemented using the Classic Extract installation method, and re-enable supplemental logging using the CDC Extract installation binaries, as the calls to enable `TRANDATA` are different between the two versions, and the implementation of `TRANDATA` for Classic Extract is not supported by the CDC Extract.

Follow these general guidelines to remove and re-enable supplemental logging. Special consideration and planning should be involved if migrating from Classic to CDC Extract in a production system. The information provided here does not cover all requirements and is only offered as general requirements regarding supplemental logging:

1. Ensure that the Classic Extract has processed all remaining data in the logs and can be gracefully stopped.

2. Do one of the following, depending on how Extract was running in relation to other replication or CDC components:

- 1. If Extract was *not* running concurrently with SQL Server transactional replication or a non-Oracle CDC configuration on the same database, open a query session in Management Studio and issue the following statement against the source database to disable and delete any CDC or replication components, and to clear the secondary truncation point.

  ```
  EXEC sys.sp_cdc_disable_db
  ```

- 2. If Extract was running *concurrently* with SQL Server transactional replication or a non-Oracle CDC configuration on the same database, run GGSCI from the Classic Extract's installation folder, login to the source database with the `DBLOGIN`, and then issue the following command for each table that is in the Extract configuration. You can use a wildcard to specify multiple table names

  ```
  DELETE TRANDATA owner.table
  ```

  ```
  DELETE TRANDATA owner.*
  ```

3. Delete any heartbeat table entries if one was installed.

```
DELETE HEARTBEATTABLE
```

4. Using the Oracle GoldenGate CDC Extract installation binaries, follow the steps listed in Preparing the Database for Oracle GoldenGate — CDC Capture to re-enable supplemental logging and other necessary components, and re-add the heartbeat table.

30

Requirements Summary for Capture and Delivery of Databases in an Always On Availability Group

Oracle GoldenGate for SQL Server supports capture from a primary replica or a read-only, synchronous mode secondary replica of an Always On Availability Group, and delivery to the primary replica.

When capturing from either a primary or a secondary replica in an Always On Availability Group, it is important to understand that the capture process must only read hardened transactions from the log, and that there be no potential for data loss between any replica database that Oracle GoldenGate is or will capture from.

Topics:

Database Connection

For both Extract and Replicat, it is recommended to create a System DSN that uses the Always On Availability Group Listener for the connection.

- For the Replicat, connecting to the Listener allows the Replicat to reconnect if the primary replica performs a failover to a new instance, without having to manually edit the DSN settings to point to the new primary.
- For the Extract connecting to the Listener not only allows reconnecting to the primary without editing the DSN to point to the new instance, but also provides the optional ability to run the Extract's data extraction stored procedures, against a read-only secondary.
- For both Extract and Replicat connected to an Always On environment, use the `AUTORESTART` parameter for the Manager, to restart the processes after a failover.
- To route the Extract's data extraction queries to a read-only secondary, ensure that the DSN connection uses the Listener, that you have one or more read-only secondary replicas that are configured to handle read-only routing, and that the Extract runs with the `TRANLOGOPTIONS ALWAYSONREADONLYROUTING` parameter.
  - Ensure that the Application Intent field of the DSN configuration is set to `READWRITE` and not `READONLY`
  - Refer to the following Microsoft documentation on how to configure read-only routing: https://docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/configure-read-only-routing-for-an-availability-group-sql-server?view=sql-server-2017

Supplemental Logging

Supplemental logging must be enabled by normal means (`ADD TRANDATA`) using GGSCI connected to the primary replica and not against a secondary replica.

- Create a DSN to the primary replica, or to the Always On Availability Group Listener, to connect using `DBLOGIN` to run `ADD TRANDATA` from GGSCI.
- The login used to enable supplemental logging must have `sysadmin` membership of the primary replica instance.
- When enabling supplemental logging against the primary replica database, the SQL Server Change Data Capture job does not automatically get created on any secondary replicas. Upon failover from a primary to a secondary, you must manually create the SQL Server Change Data Capture job and the Oracle CDC Cleanup job if in use, on the new primary replica.

  ```
  EXECUTE sys.sp_cdc_add_job N'capture
  ```

  – When creating the SQL Server CDC Capture job on the new primary, the default configuration settings are put in place. So if you have previously modified the default values on the former primary replica, you need to run `sys.sp_cdc_change_job` on the new primary and set the values accordingly.

> **Note:**
>
> Consult the Microsoft documentation on how to enable the CDC Capture job for AlwaysOn Secondary Replicas for more information:.

Operational Requirements and Considerations

- When an instance is no longer the primary instance but has the SQL Server CDC Capture job installed, the job ceases to run after some time and does not attempt to restart. Upon failover back to that instance, the job does not automatically start, so it must be manually started.
- If secondary replica databases are not in sync with the primary replica database, the CDC capture job will not advance in the log, and therefore no records will be captured by an Extract, until such time that the primary and secondary replicas are synchronized. See this article from Microsoft for more details:

  https://docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/replicate-track-change-data-capture-always-on-availability?view=sql-server-2017

> **Note:**
>
> When capturing from either a primary or a secondary replica in an Always On Availability Group, it is important to understand that the capture process must only read hardened transactions from the log, and that there be no potential for data loss between any replica database that Oracle GoldenGate is or will capture from.

- When running an Extract from a middle tier Windows or Linux server, set the middle tier server's date, time, and time zone to the same as that of the primary replica.
- Upon failover from a primary to a secondary replica, reinstall the Oracle GoldenGate CDC Cleanup job on the new primary by re-running the `ogg_cdc_cleanup_setup.bat` file with the `createJob` option.
- If Extract is configured to capture from a readable secondary replica, but not configured with read-only routing, the SQL Server CDC Capture job must be created against the secondary replica prior to starting the Extract, as the Extract will check if the job exists. To create the SQL Server CDC Capture job, any potential secondary that will have an Extract connected to it, must at some point be set to a writable Primary database and then follow the steps above, under supplemental logging, to manually add the SQL Server CDC Capture job.
- If uninstalling Oracle GoldenGate and disabling Change Data Capture on a database that is part of an Always On availability group, follow the extra steps provided in Disabling Change Data Capture.

Part VIII
Using Oracle GoldenGate for Teradata

Oracle GoldenGate for Teradata supports delivery of initial load and transactional data for supported Teradata database versions.

Oracle GoldenGate for Teradata supports the mapping, filtering, and transformation of data, unless noted otherwise in this document, as well as replicating data derived from other source databases supported by Oracle GoldenGate, into Teradata databases.

This part describes tasks for configuring and running Oracle GoldenGate for Teradata.

Topics

31
Understanding What's Supported for Teradata

This chapter contains information on database and table features supported by Oracle GoldenGate.

Topics:

Supported Teradata Data Types

The following table shows the Teradata data types that Oracle GoldenGate supports. Any limitations or conditions that apply follow this table.

Data type	v15.x	v16.x
BLOB	Yes	Yes
BYTEINT	Yes	Yes
VARBYTE	Yes	Yes
BIGINT	Yes	Yes
BYTEINT	Yes	Yes
DATE	Yes	Yes
DECIMAL - 18 and under	Yes	Yes
DECIMAL - 19 to 38	Yes	Yes
DOUBLE PRECISION	Yes	Yes
FLOAT	Yes	Yes
INTEGER	Yes	Yes
NUMERIC - 18 and under	Yes	Yes
NUMERIC - 19 to 38	Yes	Yes
REAL	Yes	Yes
SMALLIINT	Yes	Yes
TIME	Yes	Yes
TIMESTAMP	Yes	Yes
INTERVAL	Yes	Yes
INTERVAL DAY	Yes	Yes
INTERVAL DAY TO HOUR	Yes	Yes
INTERVAL DAY TO MINUTE	Yes	Yes

Data type	v15.x	v16.x
INTERVAL DAY TO SECOND	Yes	Yes
INTERVAL HOUR	Yes	Yes
INTERVAL HOUR TO MINUTE	Yes	Yes
INTERVAL HOUR TO SECOND	Yes	Yes
INTERVAL MINUTE	Yes	Yes
INTERVAL MINUTE TO SECOND	Yes	Yes
INTERVAL MONTH	Yes	Yes
INTERVAL SECOND	Yes	Yes
INTERVAL YEAR	Yes	Yes
INTERVAL YEAR TO MONTH	Yes	Yes
CHAR	Yes	Yes
CLOB	Yes	Yes
CHAR VARYING	Yes	Yes
LONG VARCHAR	Yes	Yes
VARCHAR	Yes	Yes
GRAPHIC	Yes	Yes
LONG VARGRAPHIC	Yes	Yes
VARGRAPHIC	Yes	Yes
PERIOD (DATE)	Yes	Yes
PERIOD (TIME)	Yes	Yes
PERIOD (TIMESTAMP)	Yes	Yes
UDT	Yes	Yes

Limitations of Support for Numeric Data Types

When replicating these data types from a different type of database to Teradata, truncation can occur if the source database supports a higher precision that Teradata does.

The support of range and precision for floating-point numbers depends on the host machine. In general, the precision is accurate to 16 significant digits, but you should

review the database documentation to determine the expected approximations. Oracle GoldenGate rounds or truncates values that exceed the supported precision.

Limitations of Support for Single-byte Character Data Types

Single-byte character types are fully supported within a single-byte Latin character set between other databases and Teradata. A `VARCHAR` or `CHAR` column cannot have more than 32k-1 bytes. If using UTF-16, this is 16k-2 characters.

Conditions and Limitations of Support for Multi-byte Character Data

Conditions and limitations of support for multi-byte character data are as follows:

- Install Oracle GoldenGate on a Windows or Linux replication server.
- Use the Teradata ODBC driver version 12.0.0.x or later.
- Do not use filtering, mapping, and transformation for multi-byte data types.
- A `CHAR` or `VARCHAR` column cannot contain more than 32k-1 bytes. If using UTF-16, these columns cannot contain more than 16k-2 characters.
- Set the ODBC driver to the UTF-16 character set in the initialization file.
- When creating Replicat groups, use the `NODBCHECKPOINT` option with the `ADD REPLICAT` command. The Replicat database checkpointing feature does not support an ODBC driver that is set to the UTF-16 character set. Checkpoints will be maintained in the checkpoint file on disk.

Limitations of Support for Binary Data Types

No limitations. These data types are supported between other source databases and Teradata targets.

Limitations of Support for Large Object Data Types

The following are limitations of support for large object data types.

- To replicate large objects from other databases to Teradata, use Teradata ODBC driver version 12.0 or higher on the target system. The target must support large objects that are delivered by ODBC.
- Enable the `UseNativeLOBSupport` flag in the ODBC configuration file. See the Teradata ODBC documentation.

Limitations of Support for Date Data Types

The following are limitations of support for date data types:

- `DATE`, `TIME`, and `TIMESTAMP` are fully supported when replicated from a different type of source database to Teradata.
- `TIME with TIMESZONE`, `TIMESTAMP with TIMEZONE`, and `INTERVAL` are not supported from a different type of source database to Teradata.
- Oracle GoldenGate supports timestamp data from 0001/01/03:00:00:00 to 9999/12/31:23:59:59. If a timestamp is converted from GMT to local time, these limits

also apply to the resulting timestamp. Depending on the timezone, conversion may add or subtract hours, which can cause the timestamp to exceed the lower or upper supported limit.

- Oracle GoldenGate does not support negative dates.

Limitations of Support for IDENTITY Data Types

`IDENTITY` must be configured as `GENERATED BY DEFAULT AS IDENTITY` on the target to enable the correct value to be inserted by Replicat.

Supported Objects and Operations for Teradata

This section lists the data operations and database objects that Oracle GoldenGate supports.

- Oracle GoldenGate supports the maximum number of columns per table that is supported by the database.
- Truncating operations are supported with the use of the `GETTRUNCATES` parameter with Oracle GoldenGate 12.2.*x* and greater.
- Limitations on Automatic Heartbeat Table support are as follows:
  - The `ALTER HEARTBEATTABLE` command is not supported and if used is ignored.
  - The `ADD HEARTBEATTABLE` command with the `FREQUENCY`, `PURGE_FREQUENCY`, or `RETENTION_TIME` option is not supported. When any of these options are specified with the `ADD HEARTBEATTABLE` command, a warning is displayed that the option is ignored.
  - Since Teradata does not have any internal event/job schedulers, automatic purging of heartbeat history data does not occur. You need to explicitly delete or truncate records periodically from the heartbeat history table.

Non-Supported Operations for Teradata

This section lists the data operations that Oracle GoldenGate does not support.

- Extract (capture)
- DDL

32
Preparing the System for Oracle GoldenGate

This chapter contains guidelines for preparing the database and the system to support Oracle GoldenGate. This chapter contains the following sections:
Topics:

Preparing Tables for Processing

The following table attributes must be addressed in an Oracle GoldenGate environment.

Disabling Triggers and Cascade Constraints

Disable triggers, cascade delete constraints, and cascade update constraints on target Teradata tables. Oracle GoldenGate replicates DML that results from a trigger or cascade constraint. If the same trigger or constraint gets activated on the target table, it becomes redundant because of the replicated version, and the database returns an error. Consider the following example, where the source tables are `emp_src` and `salary_src` and the target tables are `emp_targ` and `salary_targ`.

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

Ensuring Row Uniqueness for Tables

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

Unless a `KEYCOLS` clause is used in the `TABLE` or `MAP` statement, Oracle GoldenGate selects a row identifier to use in the following order of priority:

1. Primary key

2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. If none of the preceding key types exist (even though there might be other types of keys defined on the table) Oracle GoldenGate constructs a pseudo key of all columns that the database allows to be used in a unique key, excluding those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration.

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

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

- Using KEYCOLS to Specify a Custom Key

Using KEYCOLS to Specify a Custom Key

If a table does not have an applicable row identifier, or if you prefer that identifiers are not used, you can define a substitute key, providing that the table has columns that always contain unique values. You define this substitute key by including a `KEYCOLS` clause within the Extract `TABLE` parameter and the Replicat `MAP` parameter. The specified key overrides any existing primary or unique key that Oracle GoldenGate finds.

ODBC Configuration for Teradata

Configure ODBC on each target system including the creation of a data source name (DSN). A DSN stores information about how to connect to the database. See the *ODBC Driver for Teradata User Guide* for complete information and setup steps:

https://docs.teradata.com/search/books?filters=prodname~%2522ODBC+Driver+for+Teradata%2522&content-lang=en-US

Database User for Oracle GoldenGate Processes for Teradata

Follow these requirements for the database user for Oracle GoldenGate processes:

- Create a database user that is dedicated to Oracle GoldenGate. It can be the same user for all of the Oracle GoldenGate processes that must connect to a database:

  - Replicat (target database)
  - The `DEFGEN` utility (target database)
- To preserve the security of your data, and to monitor Oracle GoldenGate processing accurately, do not permit other users, applications, or processes to log on as, or operate as, the Oracle GoldenGate database user.
- For Oracle GoldenGate to replicate to a target Teradata database, grant `SELECT`, `INSERT`, `UPDATE`, and `DELETE` on all of the target tables to the Replicat database user.

33
Configuring Oracle GoldenGate

Learn about the prerequisites and tasks for configuring Oracle GoldenGate Replicat for Oracle TimesTen database.
Topics:

Creating a Checkpoint Table

Replicat maintains its checkpoints in a checkpoint table in the Teradata target database (the database where you use `DBLOGIN`). Each checkpoint is 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.

Use the following GGSCI command on the target system, to create the Replicat checkpoint table.

```
ggsci> ADD CHECKPOINTTABLE schema.chkptabl Successfully created checkpoint table schema.chkptabl
```

For more information about creating a checkpoint table, see *Administering Oracle GoldenGate*.

Configuring Oracle GoldenGate Replicat

This section highlights the basic Replicat parameters that are required for most target database types. Additional parameters may be required, see the Oracle GoldenGate installation and configuration documentation for your target database and the *Reference for Oracle GoldenGate*.

Perform these steps on the target replication server or target database system.

1. Configure the Manager process according to the instructions in *Administering Oracle GoldenGate*.
2. In the Manager parameter file, use the `PURGEOLDEXTRACTS` parameter to control the purging of files from the local trail.
3. Create a Replicat checkpoint table. There are multiple options for this purpose, see *Administering Oracle GoldenGate*.
4. Create a Replicat group. For documentation purposes, this group is called `rep`.

   ```
   ADD REPLICAT rep, EXTTRAIL remote_trail, CHECKPOINTTABLE owner.table
   ```

   Use the `EXTTRAIL` argument to link the Replicat group to the remote trail that you specified for the data pump on the source server.

5. Use the `EDIT PARAMS` command to create a parameter file for the Replicat group. Include the parameters shown in Example 33-1 plus any others that apply to your database environment.

Example 33-1 Parameters for the Replicat Group

```
-- Identify the Replicat group:
REPLICAT rep
-- Specify database login information as needed for the database:
[TARGETDB target_dsn_name,] [USERID user id[, PASSWORD pw]]
-- Specify tables for delivery:
MAP owner.source_table, TARGET owner.target_table;
```

Additional Oracle GoldenGate Configuration Guidelines

The following are additional considerations to make once you have installed and configured your Oracle GoldenGate environment.

- Handling Massive Update and Delete Operations
- Preventing Multiple Connections
- Performing Initial Synchronization

Handling Massive Update and Delete Operations

Operations that update or delete a large number of rows will generate discrete updates and deletes for each row on the subscriber database. This could cause a lock manager overflow on the Teradata subscriber system, and thus terminate the Replicat process.

To avoid these errors, temporarily suspend replication for these operations and then perform them manually on the source and target systems. To suspend replication, use the following command, which suspends replication for that session only. The operations of other sessions on that table are replicated normally.

```
set session override replication on;

commit;
```

Preventing Multiple Connections

By default, the Replicat processes create a new connection for catalog queries. You can prevent this extra connection by using the `DBOPTIONS` parameter with the `NOCATALOGCONNECT` option.

Performing Initial Synchronization

Perform an initial synchronization of the source and target data before using Oracle GoldenGate to transmit transactional changes for the first time to configure an initial load, see *Administering Oracle GoldenGate*.

34
Common Maintenance Tasks

This chapter contains instructions for performing some common maintenance tasks when using the Oracle GoldenGate replication solution.
Topics:

Modifying Columns of a Table

To modify columns of a table:

1. Suspend activity on the source database for all tables that are linked to Oracle GoldenGate.
2. Start GGSCI.
3. In GGSCI, issue this command for the Replicat group:

   ```
   INFO REPLICAT group
   ```

4. On the `Checkpoint Lag` line, verify whether there is any Replicat lag. If needed, continue to issue `INFO REPLICAT` until lag is zero, which indicates that all of the data in the trail has been processed.
5. Stop the Replicat group.

   ```
   STOP REPLICAT group
   ```

6. Perform the table modifications on the target databases.
7. Start the Replicat process.

   ```
   START REPLICAT group
   ```

8. Allow user activity to resume on all of the source tables that are linked to Oracle GoldenGate.

Part IX

Using Oracle GoldenGate for Oracle TimesTen

Oracle GoldenGate for Oracle TimesTen supports delivery of initial load and transactional data for supported Oracle TimesTen database versions.

Oracle GoldenGate for Oracle TimesTen supports the mapping, filtering, and transformation of data, unless noted otherwise in this document, as well as replicating data derived from other source databases supported by Oracle GoldenGate, into Oracle TimesTen databases.

Topics:

35

Understanding What's Supported for Oracle TimesTen

This chapter contains information on database and table features supported by Oracle GoldenGate.

Topics:

Supported Objects and Operations for TimesTen

The following objects and operations are supported:

- Oracle GoldenGate for Oracle TimesTen supports delivery of transactional DML to user tables.
- `INSERT`, `UPDATE`, `DELETE`, and `TRUNCATE` operations are supported.

Non-Supported TimesTen Data Types and Features

The `INTERVAL` and `ROWID` data types are not supported.

Supported TimesTen Data Types

The following data types are supported for delivery, unless specifically noted in the limitations that follow:

- Binary Data Types

  ```
  (binary, varbinary)
  ```

- Character Data Types

  ```
  (char, nchar, nvarchar2, varchar2)
  ```

- Date and Time Data Types

  ```
  (date, time, timestamp, tt_date, tt_time, tt_timestamp)
  ```

- Numeric Data Types

  ```
  (binary_float, binary_double, double, float, tt_bigint, tt_integer,
  number, real, tt_smallint, tt_tinyint)
  ```

- LOB

  ```
  (blob, clob, nclob)
  ```

Limitations and Non-supported Items for Oracle TimesTen

The limitations and non-supported items for Oracle TimesTen are:

- Capture (extraction) of DML operations is not supported.
- Capture and replication of DDL (data definition language) operations is not supported.
- The support of range and precision for floating-point numbers depends on the host machine. In general, the precision is accurate to 16 significant digits, but you should review the database documentation to determine the expected approximations. Oracle GoldenGate rounds or truncates values that exceed the supported precision.
- Oracle GoldenGate supports timestamp data from 0001/01/03:00:00:00 to 9999/12/31:23:59:59. If a timestamp is converted from GMT to local time, these limits also apply to the resulting timestamp. Depending on the time zone, conversion may add or subtract hours, which can cause the timestamp to exceed the lower or upper supported limit.
- Modifying the Primary Key Column value is not supported.
- Limitations on Automatic Heartbeat Table support are as follows:
  - Oracle GoldenGate supports only Delivery on TimesTen so no mechanism is required to populate/update the heartbeat tables using any event/job schedulers.
  - The `ALTER HEARTBEATTABLE` command is not supported and if used is ignored.
  - The `ADD HEARTBEATTABLE` command with the `FREQUENCY`, `PURGE_FREQUENCY`, or `RETENTION_TIME` option is not supported. When any of these options are specified with the `ADD HEARTBEATTABLE` command, a warning is displayed that the option is ignored.
  - Since TimesTen does not have any internal event/job schedulers, automatic purging of heartbeat history tables *cannot* occur. As such, you should explicitly drop or truncate the corresponding heartbeat objects to suit your environment.

36
Preparing the System for Oracle GoldenGate

This chapter contains guidelines for preparing the system to support Oracle GoldenGate:
Topics:

Choosing the Oracle TimesTen Connectivity Type

Oracle TimesTen supports two distinct connectivity types for tools, utilities and applications; Direct mode and Client-Server mode.

Direct mode
Direct mode is a highly optimised local connectivity mechanism that eliminates interprocess communication (IPC) between the application and the database. It provides low latency and high throughput with low overhead. With Direct mode, the client application and the database must reside in the same host.

Client-Server mode
Client-Server mode is a traditional TCP/IP based connection mechanism. In this mode, the client application may reside in the same host as the database but more commonly it will execute on a different host and connect over the network. Client-Server mode has lower performance than Direct mode due to additional overhead and network round trips.
Oracle GoldenGate supports both types of connectivity for Oracle TimesTen, so you can use whichever is most appropriate, based on your requirements.

Setting the Environment Variables

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

Example of setting the bash shell environment for a TimesTen instance homed in `/instancepath/tt181`:

```
source /instancepath/tt181/bin/ttenv.sh
```

Although it's possible to set the required environment variables manually, it is not recommended. Using the script:

- Ensures that all the necessary environment variables (there are several) are correctly set.
- Insulates you from the introduction of new variables in future Oracle TimesTen releases.

Configuring the TimesTen ODBC Connectivity

Oracle GoldenGate for TimesTen connects to TimesTen using the ODBC API (TimesTen's native API). ODBC connectivity defines the concept of a Data Source Name (DSN). A DSN is a logical name which applications use to specify the parameters to be used for connecting to a target database.

When using Oracle GoldenGate for TimesTen, you will specify the DSN of the target TimesTen database in various Oracle GoldenGate configuration settings, such as the `SOURCEDB` clause of the `DBLOGIN` command. For example:

```
DBLOGIN SOURCEDB database, USERIDALIAS useralias
```

Here, the value given for `database` will be the DSN of the target TimesTen database.

When using the Direct mode connectivity, connections must reference a *`server DSN`* defined in the `sys.odbc.ini` file of the Oracle TimesTen instance that hosts the database (the server instance).

When using the Client-Server mode, connections must reference a *`client DSN`* defined in the `sys.odbc.ini` file of either the Oracle TimesTen instance that manages the database (the server instance) or, more commonly, in the `sys.odbc.ini` of an Oracle TimesTen client instance, such as an Oracle GoldenGate hub server.

For information on defining Oracle TimesTen server and client DSNs, refer to TimesTen In-Memory Database Operations Guide.

Here is an example of the `sys.odbc.ini` entries to define a client DSN (`myttdbcs`) that connects to a database identified by the server DSN `myttdb` located on the host `tthost1.mydomain.com`. The TimesTen server's default listener port on that host is 6625.

```
[ODBC Data Sources]
myttdbcs=TimesTen 18.1 Client Driver
[myttdbcs]
TTC_SERVER=tthost1.mydomain.com/6625
TTC_SERVER_DSN=myttdb
ConnectionCharacterSet=AL32UTF8
```

Preparing Tables for Processing

This section describes the table attributes you must address in an Oracle GoldenGate environment with TimesTen.

Topics:

- Removing ON DELETE CASCADE Contraints
- Ensuring Row Uniqueness for Tables

Removing ON DELETE CASCADE Contraints

If a target table in Oracle TimesTen has a foreign key, which specifies the `ON DELETE CASCADE` clause, and if the table that is the target of that foreign key is also a target for Oracle GoldenGate replication then you must remove the `ON DELETE CASCADE` clause from the foreign key definition to avoid errors.

Consider the following example, where the source tables are `emp_src` and `salary_src` and the target tables are `emp_targ` and `salary_targ`.

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

Oracle TimesTen does not support the disabling of foreign key constraints or on delete cascade constraints. To remove the `ON DELETE CASCADE`, you must either drop the table and recreate it without the `ON DELETE CASCADE` clause or you can use `ALTER TABLE` to remove the foreign key constraint and recreate it without the `ON DELETE CASCADE` clause.

Ensuring Row Uniqueness for Tables

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

Unless a `KEYCOLS` clause is used in the `TABLE` or `MAP` statement, Oracle GoldenGate selects a row identifier to use in the following order of priority:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. If none of the preceding key types exist (even though there might be other types of keys defined on the table) Oracle GoldenGate constructs a pseudo key of all columns that the database allows to be used in a unique key, excluding those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration.

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

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

Configuring Oracle GoldenGate

Learn about the prerequisites and tasks for configuring Oracle GoldenGate Replicat for Oracle TimesTen database.
Topics:

Configuring Oracle GoldenGate Replicat

This section describes the Replicat parameters that are required for most target database types. For additional parameters that may be required, see the Oracle GoldenGate installation and configuration documentation for your target database and the *Reference for Oracle GoldenGate*.

Perform these steps on the target replication server or target database.

1. Configure the Manager process according to the instructions in *Administering Oracle GoldenGate*.
2. In the Manager parameter file, use the `PURGEOLDEXTRACTS` parameter to control the purging of files from the local trail.
3. Create a Replicat checkpoint table. There are multiple options for this purpose, see *Administering Oracle GoldenGate*.

   ```
   DBLOGIN SOURCEDB myttdbcs USERIDLIAS useralias
   ADD CHECKPOINTTABLE owner.oggcheckpointtable
   ```

4. Create a Replicat. For documentation purposes, this Replicat is called `reptt`.

   ```
   ADD REPLICAT reptt, EXTTRAIL ./dirdat/remote_trail, CHECKPOINTTABLE
   owner.oggcheckpointtable
   ```

   Use the `EXTTRAIL` argument to link the Replicat to the remote trail that you specified with the Data Pump Extract on the source Oracle GoldenGate installation.
5. Use the `EDIT PARAMS` command to create a parameter file for the Replicat group. Include the parameters shown in Example 33-1 plus any others that apply to your database environment.

Example 36-1 Parameters for the Replicat Group

```
REPLICAT reptt
-- Specify database login information as needed for the database:
TARGETDB myttdbcs, USERIDALIAS useralias
```

```
-- Specify tables for delivery:
MAP owner.sourcetable, TARGET owner.targettable;
```

Additional Oracle GoldenGate Configuration Guidelines

The following are additional considerations to make once you have installed and configured your Oracle GoldenGate environment.

Performing Initial Synchronization

Perform an initial synchronization of the source and target data before using Oracle GoldenGate to transmit transactional changes for the first time to configure an initial load. See Initial Synchronization in *Administering Oracle GoldenGate*.