Oracle® Fusion Middleware

Using Oracle GoldenGate for Heterogeneous Databases

12*c* (12.3.0.1)
E88786-09
November 2018

Oracle Fusion Middleware Using Oracle GoldenGate for Heterogeneous Databases, 12*c* (12.3.0.1)

E88786-09

Primary Author: Oracle Corp.

Contents

Preface

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

Topics:

Audience

Using Oracle GoldenGate for Heterogeneous Databases is intended for DBA 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

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

Oracle GoldenGate A-Team Chronicles

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I

What is Oracle GoldenGate for Heterogeneous Databases?

Oracle GoldenGate is a comprehensive software package for real-time data capture and replication in heterogeneous IT environments.

The product set enables high availability solutions, real-time data integration, transactional change data capture, data replication, transformations, and verification between operational and analytical enterprise systems. Oracle GoldenGate 12c brings extreme performance with simplified configuration and management, support for cloud environments, expanded heterogeneity, and enhanced security.

You can use the following supported heterogeneous databases with Oracle GoldenGate.

- DB2 LUW
- DB2 for i
- DB2 for z/OS
- MySQL
- SQL Server
- Teradata

Each database that Oracle GoldenGate supports has it's own requirements and configuration. This book is divided into parts so that you can easily find information that is relevant to your environment. See *Installing Oracle GoldenGate* for system requirements and installation details for each of these databases.

Part II
Using Oracle GoldenGate with DB2 LUW

With the Oracle GoldenGate for DB2 LUW databases, you can replicate data to and from the supported DB2 LUW versions or between a DB2 LUW database and a database of another type. Oracle GoldenGate for DB2 LUW supports data filtering, mapping, and transformation unless noted otherwise in this documentation.

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

1

Understanding What's Supported for DB2 LUW

This chapter contains support information for Oracle GoldenGate on DB2 LUW databases.

Topics:

1.1 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.
- `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(9)`. Extract also captures `TIMESTAMP(10)` through `TIMESTAMP(12)`, but it truncates the data to nanoseconds (maximum of nine digits of fractional time) and issues a warning to the error log. Replicat truncates timestamp data from other sources to nanoseconds when applying it to `TIMESTAMP(10)` through `TIMESTAMP(12)` in a DB2 LUW target.
- 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.

1.2 Non-Supported DB2 LUW Data Types

The non-supported DB2 LUW data types are:

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

1.3 Supported Objects and Operations for DB2 LUW

The supported objects and operations for DB2 LUW are:

- Oracle GoldenGate supports the maximum number of columns and column size per table that is supported by the database.
- `TRUNCATE TABLE` for DB2 LUW version 9.7 and later.
- Multi Dimensional Clustered Tables (MDC) for DB2 LUW 9.5 and later.
- 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, the options`COMPRESS YES STATIC` and `COMPRESS YES ADAPTIVE` are supported. To support the use of `COMPRESS YES` in DB2 LUW versions 9.7 and earlier, you must use the `TRANLOGOPTIONS` parameter with the `ALLOWTABLECOMPRESSION` option, and the compressed table must not contain LOBs.
- The extended row size feature is enabled by default. It is supported with a workaround by using the `FETCHCOLS` option. 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 the column value. If the value is out of row and `FETCHCOLS` is *not* specified then the Extract process abends to prevent any data loss. If you do not want to use this feature, set the `extended_row_size` parameter to `DISABLE`.
- Temporal tables with DB2 LUW 10.1 FixPack 2 and greater are supported. This is the default for the Replicat process.
- Limitations on Automatic Heartbeat Table support are as follows:

- 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 the cron format so the Oracle GoldenGate input value to the 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
     ```

1.4 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:**

1.5 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:

2.1 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.

2.1.1 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.
   ```

2.1.2 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`.

2.2 Preparing Tables for Processing

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

- Disabling Triggers and Cascade Constraints
- Assigning Row Identifiers
- Preventing Key Changes
- Enabling Change Capture
- Maintaining Materialized Query Tables

2.2.1 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`.

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.

2.2.2 Assigning Row Identifiers

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.

- How Oracle GoldenGate Determines the Kind of Row Identifier to Use
- Using KEYCOLS to Specify a Custom Key

2.2.2.1 How Oracle GoldenGate Determines the Kind of Row Identifier to Use

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.

2.2.2.2 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.

2.2.3 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.

2.2.4 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.

2.2.5 Maintaining Materialized Query Tables

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

The following are the rules for configuring these tables:

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

2.3 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.

2.4 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 LSN 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*.

2.5 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:

3.1 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.

3.2 Where to Get More Information

See Administering Oracle GoldenGate for Windows and UNIX 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
- Administrative topics

3.3 Configuring the Primary Extract

These steps configure the primary Extract to capture transaction data from a source DB2 LUW database 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
   LOGALLSUPCOLS
   ENCRYPTTRAIL AES192
   EXTTRAIL /ggs/dirdat/lt
   TABLE hr.*;
   ```

Table 3-1 Basic Parameters for Primary Extract

Parameter	Description
`EXTRACT group`	*group* is the name of the Extract group. For more information, see *Reference for Oracle GoldenGate*.
`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. For more information, see Database User for Oracle GoldenGate Processes.
`ENCRYPTTRAIL algorithm`	Encrypts the local trail. For more information about Oracle GoldenGate trail encryption options, see *Administering Oracle GoldenGate* .
`EXTTRAIL pathname`	Specifies the path name of the local trail to which the primary Extract writes captured data for temporary storage.

Table 3-1 (Cont.) Basic Parameters for Primary Extract

Parameter	Description
`TABLE schema.object;`	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. See *Administering Oracle GoldenGate* for information about how to specify object names with and without wildcards. Note that 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. See *Reference for Oracle GoldenGate* for more information about usage and syntax. For more information and for additional `TABLE` options that control data filtering, mapping, and manipulation, see *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. For a list of parameters and links to their detailed reference, see *Reference for Oracle GoldenGate*.
4. Save and close the file.

3.4 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. See Table 3-2 for descriptions.

   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.*;
   ```

Table 3-2 Basic Parameters for a Data-pump Extract

Parameter	Description
`EXTRACT group`	*group* is the name of the data pump Extract. For more information, see *Oracle Fusion Middleware Reference for Oracle GoldenGate for Windows and UNIX*.

Table 3-2 (Cont.) Basic Parameters for a Data-pump Extract

Parameter	Description
`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. For more information, see Database User for Oracle GoldenGate Processes.
`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. For additional options and encryption details, see *Oracle Fusion Middleware Reference for Oracle GoldenGate for Windows and UNIX*.
`RMTTRAIL pathname`	Specifies the path name of the remote trail. For more information, see *Oracle Fusion Middleware Reference for Oracle GoldenGate for Windows and UNIX*.
`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. See *Oracle Golden Gate Administering Oracle GoldenGate for Windows and UNIX* for information about how to specify object names with and without wildcards. Note that 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. See *Oracle Fusion Middleware Reference for Oracle GoldenGate for Windows and UNIX* for more information about usage and syntax. For more information and for additional `TABLE` options that control data filtering, mapping, and manipulation, see *Oracle Fusion Middleware Reference for Oracle GoldenGate for Windows and UNIX*.

3. Enter any optional Extract parameters that are recommended for your configuration. You can edit this file at any point before starting processing by using the `EDIT PARAMS` command in GGSCI. For a list of parameters and links to their detailed reference, see *Oracle Fusion Middleware Reference for Oracle GoldenGate for Windows and UNIX*.
4. Save and close the file.

3.5 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. See Table 6-3 for descriptions.

```
REPLICAT financer
TARGETDB FINANCIAL USERID ogg, PASSWORD AACAAAAAAAAAAA, BLOWFISH ENCRYPTKEY mykey
ASSUMETARGETDEFS
-- Instead of ASSUMETARGETDEFS, use SOURCEDEFS if replicating from
-- DB2 for i to a different database type, or from a DB2 for i source
-- that is not identical in definitions to a target DB2 for i database.
-- SOURCEDEFS /users/ogg/dirdef/defsfile
DISCARDFILE /users/ogg/disc
MAP hr.*, TARGET hr2.*;
```

Table 3-3 Basic Parameters for Replicat

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.
`SOURCEDEFS pathname | ASSUMETARGETDEFS`	Specifies how to interpret data definitions. Use SOURCEDEFS if the source and target tables have different definitions, such as when replicating data between dissimilar IBM for i databases or from an IBM for i database to an Oracle database. For *pathname*, specify the source data-definitions file that you created with the DEFGEN utility in "Creating a Data Definitions File". Use ASSUMETARGETDEFS if the source and target tables are all DB2 for i and have the same definitions.
`MAP owner.table, TARGET owner.table;`	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 or library name. • *table* is the name of a table or a wildcard definition for multiple tables. For supported object names, see *Administering Oracle GoldenGate*. Terminate the MAP statement with a semi-colon. 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.

- Creating a Temporal Table
- Creating a Checkpoint Table
- Configuring the Replicat Parameter File

3.5.1 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

3.5.1.1 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.

3.5.1.2 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.

3.5.1.3 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'));
```

3.5.2 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*.

3.5.3 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. See Table 3-4 for descriptions.

   Basic parameters for the Replicat group:

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

Table 3-4 Basic Parameters for Replicat

Parameter	Description
`REPLICAT group`	*group* is the name of the Replicat group.

Table 3-4 (Cont.) Basic Parameters for Replicat

Parameter	Description
`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. For more information about data definitions, see *Oracle Golden Gate Administering Oracle GoldenGate for Windows and UNIX*.
`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. For more information and for additional options that control data filtering, mapping, and manipulation, see `MAP` in *Oracle Fusion Middleware Reference for Oracle GoldenGate for Windows and UNIX*.

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. For a list of parameters and links to their detailed reference, see *Oracle Fusion Middleware Reference for Oracle GoldenGate for Windows and UNIX*.
4. Save and close the file.

3.6 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 for Windows and UNIX. This guide also contains information about:
  - Oracle GoldenGate architecture

  - Oracle GoldenGate commands
  - Oracle GoldenGate initial load methods
  - Configuring security
  - 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 for Windows and UNIX.

3.7 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.

3.8 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 III

Using Oracle GoldenGate with IBM DB2 for i

This part describes tasks for configuring and running Oracle GoldenGateon a DB2 for i database.

With Oracle GoldenGate for DB2 for i, you can:

- Map, filter, and transform transactional data changes between similar or dissimilar supported DB2 for i versions, and between supported DB2 for i versions and other supported types of databases.
- Perform initial loads from DB2 for i to target tables in DB2 for i or other databases to instantiate a synchronized replication environment.

Topics:

4

Understanding What's Supported for IBM DB2 for i

With Oracle GoldenGate on DB2 for i, you can replicate data to and from similar or dissimilar supported DB2 for i versions, or you can replicate data between a DB2 for i database and a database of another type.

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 directly 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 directly on the DB2 for i target system or on a remote Windows or Linux system. If installed on a remote system, Replicat delivers the data 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:

4.1 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

The Extract process fully supports the capture and apply of `TIMESTAMP(0)` through `TIMESTAMP(6)`. Extract also captures `TIMESTAMP(7)` through `TIMESTAMP(12)`, but it truncates the data to microseconds (maximum of six digits of fractional time) and issues a warning to the error log. Replicat truncates timestamp data from other sources to microseconds when applying it to `TIMESTAMP(7)` through `TIMESTAMP(12)` in a DB2 for i target.

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.

4.2 Non-Supported DB2 for i Data Types

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

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

4.3 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.
- `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
- Limitations on Automatic Heartbeat Table support are as follows:
  - The `ADD HEARTBEATTABLE` command creates a new file called `ogghbfreq¿` in the Oracle GoldenGate build folder. 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 from the time the `ADD HEARTBEATTABLE` command is given and 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.

4.4 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
- The Multi-Journal feature does not support multi-journal sync of a transaction across multiple journals.

4.5 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` (not supported on V5R4 and i6.1 only)

`BR`

`ASCIITOEBCDIC` and `EBCDICTOASCII`

`BINARYCHARS`

`LOBMEMORY`

`TRAILCHARSETEBCDIC`

Any encryption options that use AES encryption

4.6 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*`

Oracle GoldenGate does not allow wildcarding for library names.

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 table names, but not schema names. For instructions on wildcarding SQL names, see Specifying Object Names in Oracle GoldenGate Input in *Administering Oracle GoldenGate*.

4.7 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 `CHARSET` and `COLCHARSET` parameters to override the default behavior as follows. See CHARSET and COLCHARSET in *Reference for Oracle GoldenGate*

When `COLCHARSET PASSTHRU` is provided for a particular column, that column is not converted into Unicode, and its source character set is provided in the trail metadata, so that Replicat can correctly process the trail data in the native character set of the column improving performance on the source side. To enable the trail to contain multiple character sets, the columns should be configured in `PASSTRHU` mode. For example:

```
TABLE GGSCHEMA.TABLE1, CHARSET(PASSTHRU);
TABLE GGSCHEMA.TABLE2, COLCHARSET(PASSTHRU, COL1, COL2, COL3);
```

Where *`COL1`*, *`COL2`*, *`COL3`* are `CHAR` or `VARCHAR` fields with a valid CCSID.

For a CCSID 65535 character field (equivalent to a `BINARY` or `CHAR FOR BIT DATA` field), you can specify a column character set override with the valid character set of the column. The field will no longer been seen as a binary field by the Oracle GoldenGate processes and is seen as a normal character field. The data is converted to UTF8, and you cannot use the `PASSTHRU` capability if a `COLCHARSET` is specified to provide a specific character encoding for a binary field. The conversion to UTF-8 makes it possible for this type of override to be compatible with HP NonStop or any other platform that requires essentially ASCII compatible data from the source. The default behavior for a CCSID 65535 `CHAR` or `VARCHAR` field is not to convert the field and bind it as binary. For example:

```
TABLE GGSCHEMA.TABLE3, COLCHARSET(IBM037, COL1, COL2);
```

Where *COL1* and *COL2* are `CHAR` or `VARCHAR` fields with CCSID 65535.

You can set a column character override for any `CHAR`, `CLOB`, or `VARCHAR` column with a valid character set (not CCSID 655350 when the data contained in the column has a different character set than what you intended. For example, if there is a `CHAR` column with CCSID set as 1047, and the data contained in it is actually in CCSID 37, you can set a column override CCSID for the column in the Extract or DEFGEN parameter file so that when it processed, the processes recognize that the data is actually in CCSID 37 and not in CCSID 1047. The data is treated as CCSID 37, but converted to UTF8.

You cannot override the CHARSET and use PASSTHRU together. For example:

```
TABLE GGSCHEMA.TABLE4, COLCHARSET(IBM037, COL1);
```

Where *COL1* is a CHAR or VARCHAR field defined with CCSID 1047 and the data is actually in CCSID 37.

If the column character set of a source column is not a valid character set supported by Oracle GoldenGate ICU and you specify COLCHAR SETPASSTHRU in the Extract or DEFGEN parameter file, then the PASSTHRU behavior is ignored, and the column data is converted to Unicode. This ensures that the data is convertible on the Replicat. Since the data has no ICU representation, there is no way to indicate what character set the data is really in.

5

Preparing the System for Oracle GoldenGate

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

Topics:

5.1 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.

5.1.1 Allocating Journals to an Extract Group

One Extract process can handle up to 30 journals. 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 journals 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*.

5.1.2 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)
- Journal State (`JRNSTATE`): `*ACTIVE`
- Minimize Entry Specific Data (`MINENTDTA`): `*NONE`
- Fixed Length Data (`FIXLENDTA`): `*USR *SYSSEQ`

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(*MAXOPT2)
JRNSTATE(*ACTIVE) MINENTDTA(*NONE) FIXLENDTA(*USR *SYSSEQ)
```

> **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.

5.1.3 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.

5.2 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.

5.3 Preparing Tables for Processing

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

- Assigning Row Identifiers
- Preventing Key Changes
- Disabling Constraints on the Target
- Enabling Change Capture
- Maintaining Materialized Query Tables
- Specifying the Oracle GoldenGate Library

5.3.1 Assigning Row Identifiers

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.

- How Oracle GoldenGate Determines the Kind of Row Identifier to Use
- Using KEYCOLS to Specify a Custom Key

5.3.1.1 How Oracle GoldenGate Determines the Kind of Row Identifier to Use

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.

5.3.1.2 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.

5.3.2 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
   ```

5.3.3 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.

5.3.4 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.

   > **Note:**
   >
   > Only `BLOWFISH` encryption is supported for DB2 for i systems.

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

5.3.4.1 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.

5.3.4.2 Removing a Default Journal Specification

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

```
DEFAULTJOURNAL CLEAR
```

5.3.5 Maintaining Materialized Query Tables

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

The following are the rules for configuring these tables:

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

5.3.6 Specifying the Oracle GoldenGate Library

Before starting Oracle GoldenGate, specify the name of the Oracle GoldenGate for DB2 for i library when running the ggos400install script. This creates a link to the OGGPRCJRN *SRVPGM (service program) object that was restored to that library. If the link to the oggprcjrn service program is deleted you could just re-run the ggos400install shell script and specify the same library, or use the command "ln -s /qsys.lib/

OGG_library.lib/oggprcjrn.srvpgm oggprcjrn.srvpgm". If this link is incorrect or missing, Extract will abend.

5.4 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.

5.5 Configuring the ODBC Driver

This section applies only if you installed Replicat on a Windows or Linux system to operate remotely from the DB2 for i target. In this configuration, Replicat must connect to the target system over a database connection that is specified with ODBC. The following steps show how to install and configure ODBC to connect to the DB2 for i target system.

Configuring ODBC on Linux

Configuring ODBC on Windows

- Configuring ODBC on Linux
- Configuring ODBC on Windows

5.5.1 Configuring ODBC on Linux

To configure ODBC, you can use the graphical user interface to run the ODBC configuration utility that is supplied with your Linux distribution, or you can edit the `odbc.ini` file with the settings described in these steps. (These steps show the ODBC Administrator tool launched from the `ODBCConfig` graphical interface utility for Linux.)

1. Download and install the 32-bit or 64-bit iSeries Access ODBC driver on the remote Linux system according to the vendor documentation. The iSeries ODBC driver is supplied as a free component of iSeries Access.
2. Issue one of the following commands, depending on the driver that you want to use.

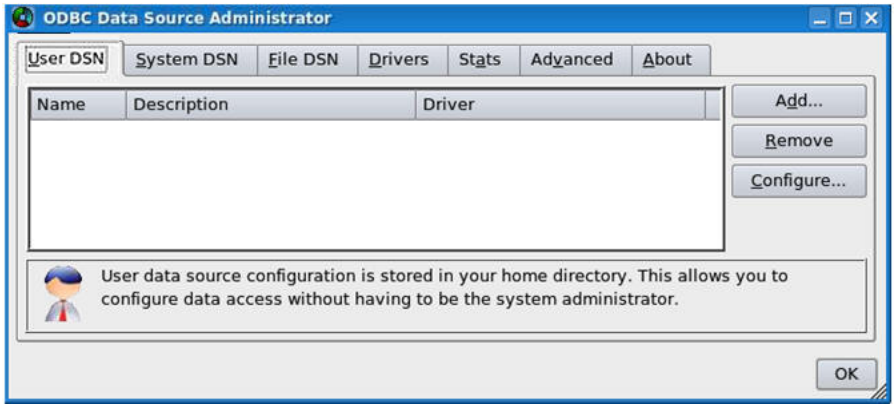
   32-bit driver:

   ```
   rpm -ivh iSeriesAccess-7.1.0-1.0.i386.rpm
   ```

   64-bit driver:

   ```
   rpm -ivh iSeriesAccess-7.1.0-1.0.x86_64.rpm
   ```
3. You can create a user DSN (a connection that is available only to the user that created it) or a system DSN (a connection that is available to all users on the system). To create a user DSN, log on to the system as the user that you will be using for the Replicat process.
4. Run the ODBC configuration utility.
5. On the initial page of the ODBC configuration tool, select the **User DSN** tab to create a user DSN or the **System DSN** tab to create a system DSN. (These steps create a user DSN; creating a system DSN is similar.)

Figure 5-1 Adding an ODBC DSN on Linux



6. On the tab you selected, click **Add**.
7. Select the appropriate iSeries Access ODBC driver, click **OK**, and then go to step 8 of these instructions. If the correct driver is not shown in the **Select the DRIVER** list, click the **Add...** button and then complete the fields as shown in Figure 3–3.

Figure 5-2 Selecting an ODBC Driver on Linux

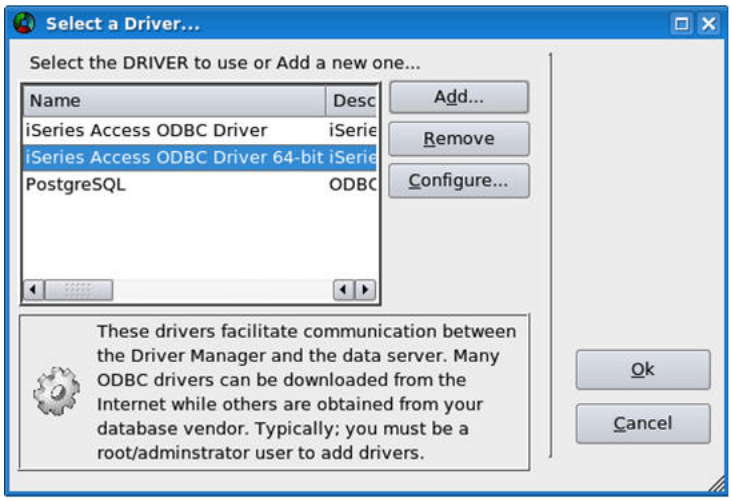
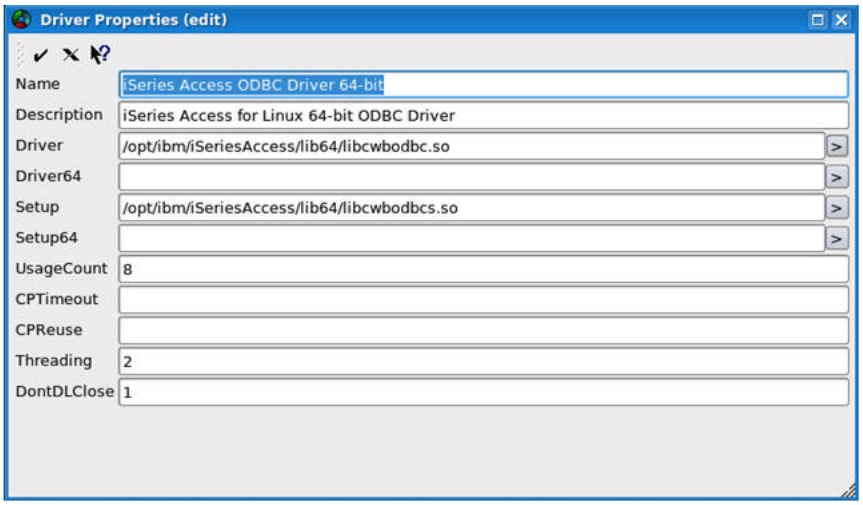


Figure 5-3 Manually Editing Driver Properties When the Driver is Not Found

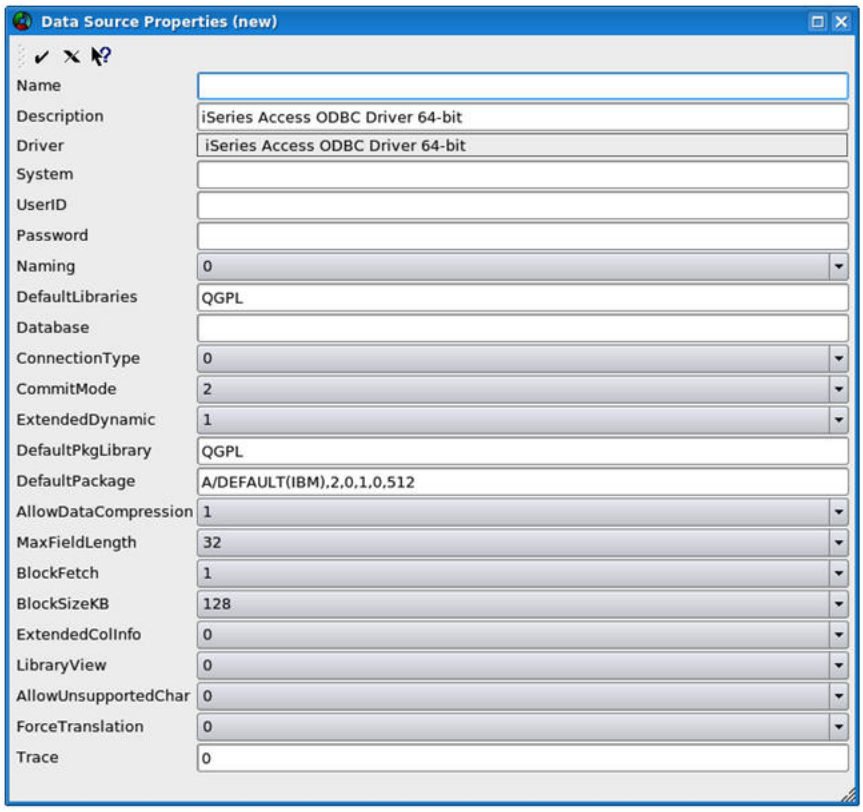


How to complete Figure 5-3 fields:

- Set **Name** to the name of the driver.
- Set **Driver** to the path where the driver is installed.
- Set **Setup** to the `libcwbodbci.so` file that is in the driver installation directory.
- Leave the other settings to their defaults.
- Click the check mark above the **Name** field to save your settings.

8. In the **Name** field of the Data Source Properties dialog, supply a one-word name for the data source. In the **System** field, enter the fully qualified name of the target DB2 for i system, for example: `sysname.company.com`. Leave the **UserID** and **Password** fields blank, to allow Replicat to supply credentials when it connects to the database. Leave the remaining fields set to their defaults, and then click the check mark above the **Name** field to save your settings.

Figure 5-4 Setting Data Source Properties



9. You are returned to the ODBC Data Source Administrator dialog. Click **OK** to exit the ODBC configuration utility.
10. To support `GRAPHIC`, `VARGRAPHIC` and `DBCLOB` types, edit the `.odbc.ini` file and add the following line.

    ```
    GRAPHIC = 1
    ```

    > **Note:**
    >
    > If you created a user Data Source Name, this file is located in the home directory of the user that created it. If you created a system DSN, this file is in `/etc/odbc.ini` or `/usr/local/etc/odbc.ini`.

11. From the Oracle GoldenGate directory on the target, run GGSCI and issue the `DBLOGIN` command to log into the target database. See *Reference for Oracle GoldenGate* for detailed syntax.

    ```
    DBLOGIN SOURCEDB database, USERID db_user [, PASSWORD pw [encryption options]]
    ```

    Where:

    - `SOURCEDB database` specifies the new Data Source Name.
    - `USERID db_user, PASSWORD pw` are the Replicat database user profile and password.
    - `encryption options` is optional password encryption.

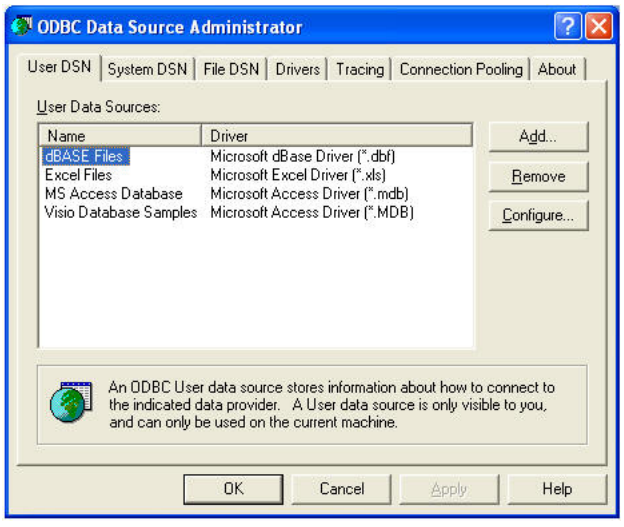
> **Note:**
>
> Only `BLOWFISH` encryption is supported for DB2 for i systems.

5.5.2 Configuring ODBC on Windows

On Windows, the ODBC Administration tool is in the Administrative Tools folder as **Data Sources (ODBC)**.

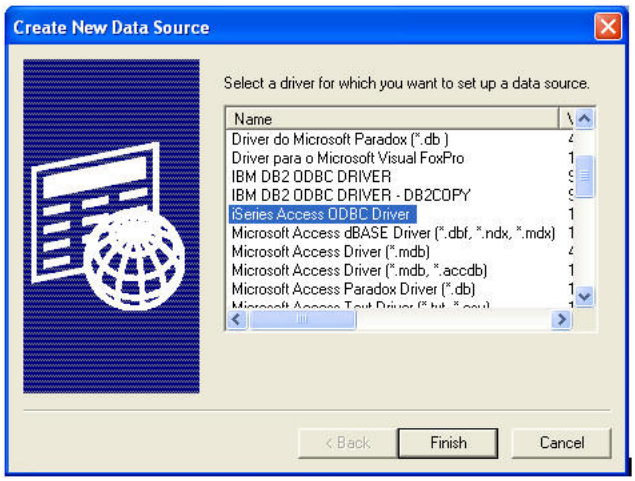
1. Download and install the 32-bit or 64-bit iSeries Access ODBC driver from the DB2 for iSeries Access package on the remote Windows system according to the vendor documentation. The iSeries ODBC driver is supplied as a free component of iSeries Access.
2. You can create a user DSN (a connection that is available only to the user that created it) or a system DSN (a connection that is available to all users on the system). To create a user DSN, log on to the system as the user that you will be using for the Replicat process.
3. From the Windows Control Panel, select **Administrative Tools**, then **Data Sources (ODBC)**.
4. On the first page of the ODBC configuration tool, select the **User DSN** tab to create a user DSN or the **System DSN** tab to create a system DSN. (These steps create a user DSN; creating a system DSN is similar.)

Figure 5-5 Adding an ODBC DSN on Windows



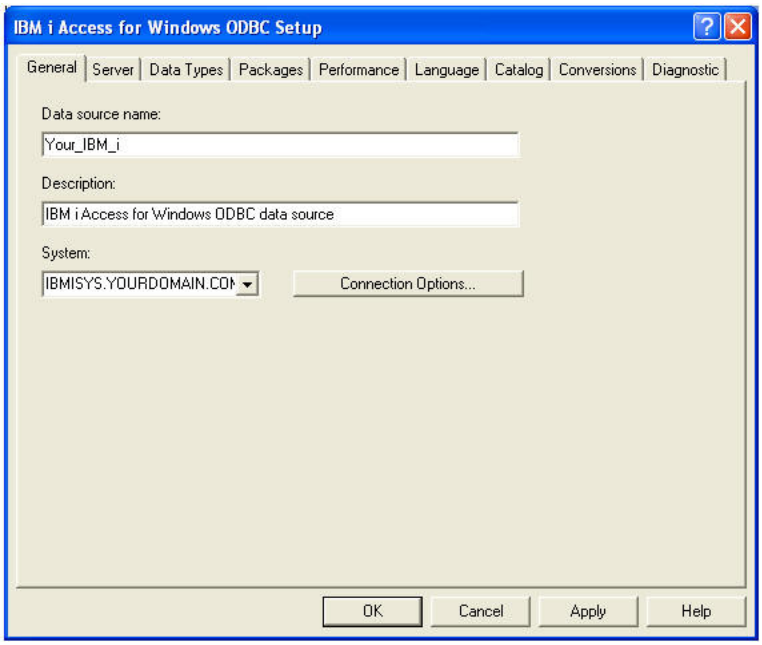
5. On the tab that you selected, click **Add**.
6. Select the appropriate iSeries Access ODBC Driver from the list of drivers, and then click **Finish**.

Figure 5-6 Selecting an ODBC Driver on Windows



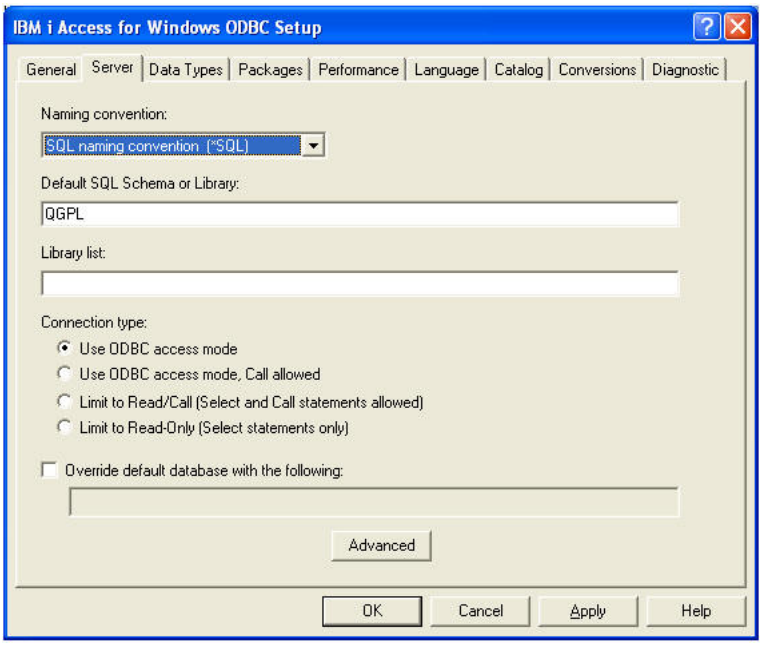
7. On the General tab of the DB2 for i Access for Windows ODBC Setup dialog, provide a name (without any spaces) in the **Data Source Name** field, add an optional description in the **Description** field, and then select the system name from the **System** selection list.

Figure 5-7 Setting General ODBC Properties on Windows



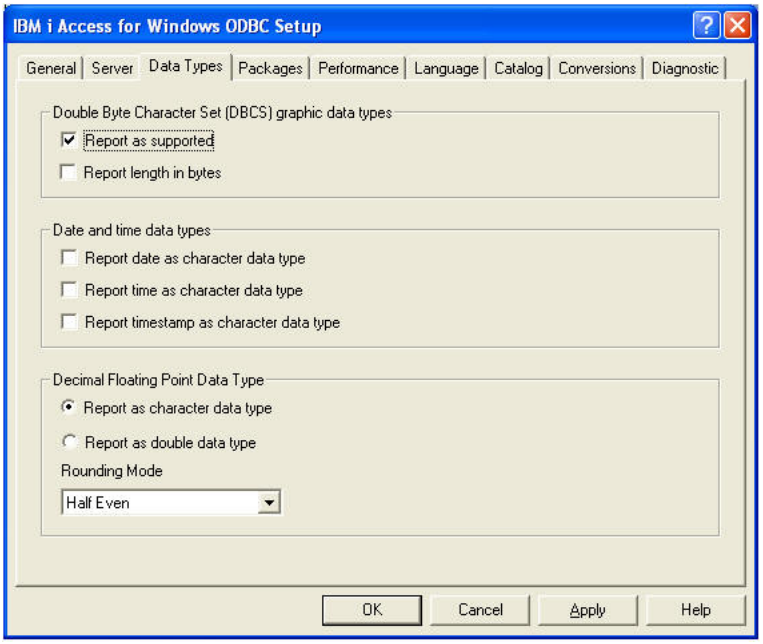
8. On the Server tab, set **Naming Convention** to `SQL Naming Convention (*SQL)`. Leave the other fields set to their defaults.

Figure 5-8 Setting Server ODBC Properties on Windows



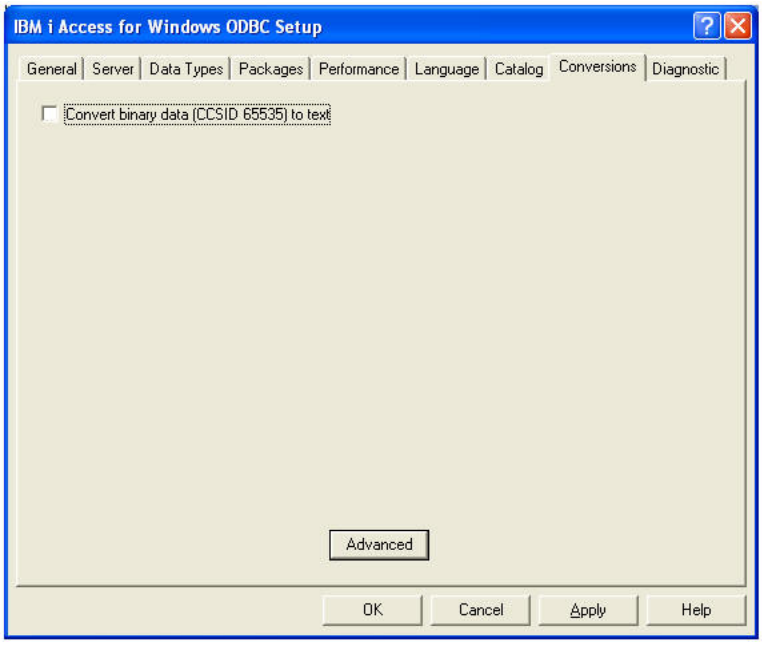
9. On the Data Types tab, select the **Report as Supported** check box under Double Byte Character Set (DBCS) graphic data types.

Figure 5-9 Setting DBCS ODBC Properties



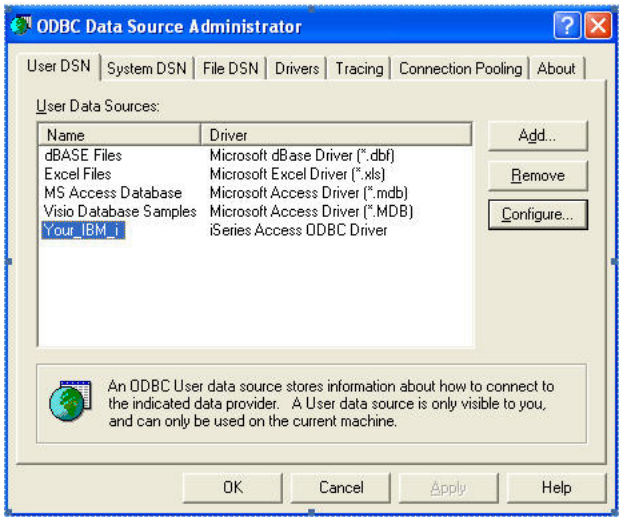
10. On the Conversions tab, clear the **Convert binary data (CCSID 65535) to text** check box.

Figure 5-10 Setting Conversion ODBC Properties



11. Click **Apply**, then **OK**. You are returned to the ODBC Data Source Administrator dialog.
12. Confirm that the new Data Source Name appears under **User Data Sources**.

Figure 5-11 Confirming New Data Source Name



13. Click **OK** to exit the ODBC configuration utility.
14. From the Oracle GoldenGate directory on the target, run GGSCI and issue the `DBLOGIN` command to log into the target database. See *Reference for Oracle GoldenGate* for detailed syntax.

    ```
    DBLOGIN SOURCEDB database, USERID db_user [, PASSWORD pw [encryption_options]]
    ```

    Where:

- `SOURCEDB` *`database`* specifies the new data source name.
- `USERID` *`db_user`*`, PASSWORD` *`pw`* are the Replicat database user profile and password.
- *`encryption_options`* is optional password encryption.

> **Note:**
>
> Only `BLOWFISH` encryption is supported for DB2 for i systems.

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:

6.1 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 is a subset of the instructions in *Administering Oracle GoldenGate*. It 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.

6.2 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.

6.3 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. For details on configuring Manager and its network connections, see *Administering Oracle GoldenGate*.

6.4 Creating a `GLOBALS` File

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

6.5 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:

- 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. For instructions, see *Administering Oracle GoldenGate*. This file must be in place on the target system before you start the Oracle GoldenGate processes for the first time.

6.6 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. Oracle GoldenGate must use Blowfish encryption on the DB2 for i platform. The standard Oracle GoldenGate encryption method of AES (Advanced Encryption Standard) is not supported by the IMB i platform. To encrypt the password, see *Administering Oracle GoldenGate*. That documentation 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.

6.7 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

6.7.1 Configuring the Primary Extract

These steps configure the primary Extract to capture transaction data from a source DB2 LUW database 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
   LOGALLSUPCOLS
   ENCRYPTTRAIL AES192
   EXTTRAIL /ggs/dirdat/lt
   TABLE hr.*;
   ```

Table 6-1 Basic Parameters for Primary Extract

Parameter	Description
`EXTRACT group`	*group* is the name of the Extract group. For more information, see *Reference for Oracle GoldenGate*.

Table 6-1 (Cont.) Basic Parameters for Primary Extract

Parameter	Description
`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. For more information, see Database User for Oracle GoldenGate Processes.
`ENCRYPTTRAIL algorithm`	Encrypts the local trail. For more information about Oracle GoldenGate trail encryption options, see *Administering Oracle GoldenGate* .
`EXTTRAIL pathname`	Specifies the path name of the local trail to which the primary Extract writes captured data for temporary storage.
`TABLE schema.object;`	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. See *Administering Oracle GoldenGate* for information about how to specify object names with and without wildcards. Note that 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. See *Reference for Oracle GoldenGate* for more information about usage and syntax. For more information and for additional `TABLE` options that control data filtering, mapping, and manipulation, see *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. For a list of parameters and links to their detailed reference, see *Reference for Oracle GoldenGate*.
4. Save and close the file.

6.7.2 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 Table 6-2 for descriptions.

   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.*;
   ```

Table 6-2 Basic Parameters for the Data Pump Extract

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 owner.table;`	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 owner.table` 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 Commands.
4. Save and close the file.

6.8 Configuring Replicat for Change Delivery to DB2 for i

These steps configure Replicat to apply data to a DB2 for i target database, operating either on the target system or on a remote Windows or 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
- Configuring Replicat

6.8.1 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.

   ```
   DEFAULTJOURNAL library_name/journal_name
   ```

   Where: *library_name* is the name of the library and journal_name is the name of the default journal.

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*.

6.8.2 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. See Table 6-3 for descriptions.

   ```
   REPLICAT financer
   TARGETDB FINANCIAL USERID ogg, PASSWORD AACAAAAAAAAAAA, BLOWFISH ENCRYPTKEY mykey
   ASSUMETARGETDEFS
   -- Instead of ASSUMETARGETDEFS, use SOURCEDEFS if replicating from
   -- DB2 for i to a different database type, or from a DB2 for i source
   -- that is not identical in definitions to a target DB2 for i database.
   -- SOURCEDEFS /users/ogg/dirdef/defsfile
   DISCARDFILE /users/ogg/disc
   MAP hr.*, TARGET hr2.*;
   ```

Table 6-3 Basic Parameters for Replicat

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.
`SOURCEDEFS pathname | ASSUMETARGETDEFS`	Specifies how to interpret data definitions. Use `SOURCEDEFS` if the source and target tables have different definitions, such as when replicating data between dissimilar IBM for i databases or from an IBM for i database to an Oracle database. For *pathname*, specify the source data-definitions file that you created with the `DEFGEN` utility in "Creating a Data Definitions File". Use `ASSUMETARGETDEFS` if the source and target tables are all DB2 for i and have the same definitions.
`MAP owner.table, TARGET owner.table;`	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 or library name. • *table* is the name of a table or a wildcard definition for multiple tables. For supported object names, see *Administering Oracle GoldenGate*. Terminate the `MAP` statement with a semi-colon. 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.

6.9 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 for Windows and UNIX. This guide also contains information about:
  - Oracle GoldenGate architecture
  - Oracle GoldenGate commands
  - Oracle GoldenGate initial load methods
  - Configuring security

  - 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 for Windows and UNIX.

6.10 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

6.10.1 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.

6.10.2 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. For more information about these commands, see *Reference for Oracle GoldenGate*.

> **Note:**
>
> Because the journals can have a transaction split among them, if a given journal is independently repositioned far into the past, the resulting latency from reprocessing the entries may cause the already-read journals to stall until the reading of the latent journal catches up.

6.11 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:

7.1 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.

7.2 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

7.3 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

7.3.1 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.

7.3.2 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*.

7.3.3 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.

7.4 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

7.4.1 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.

7.4.2 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

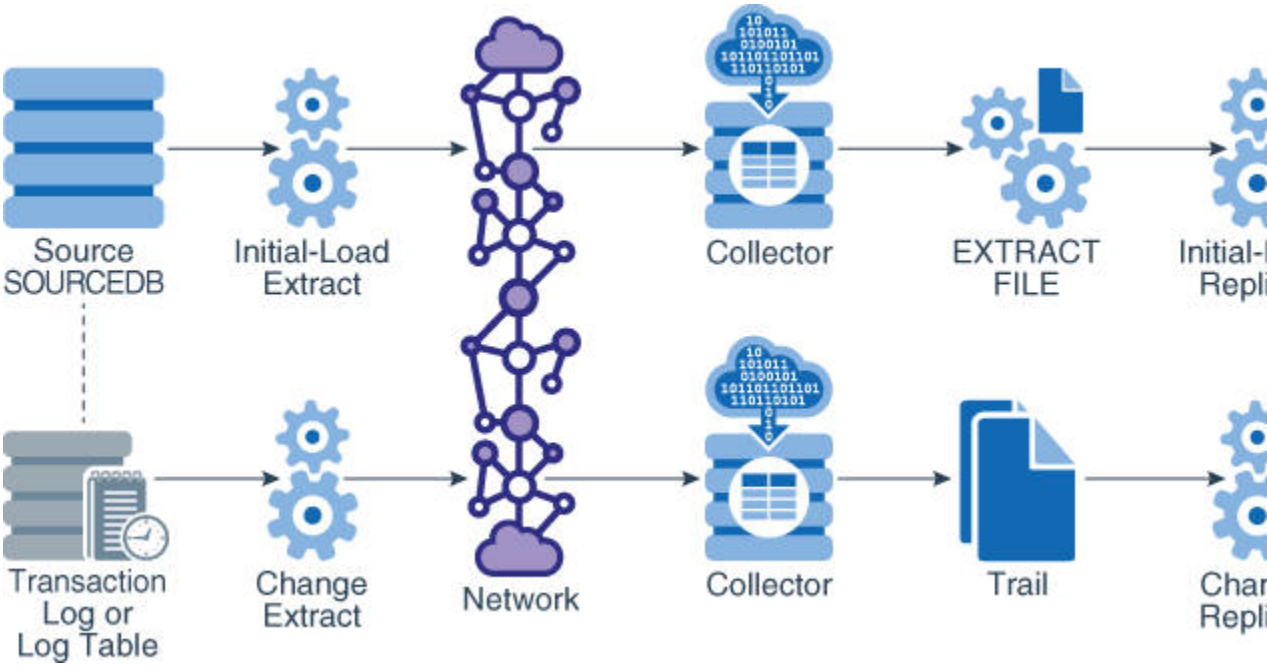
of processes. You can isolate large tables from smaller ones by using different sets of processes, or simply apportion the load across any number of process sets. To configure parallel processes correctly, see Administering Oracle GoldenGate for Windows and UNIX.

7.5 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

7.5.1 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.
`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.

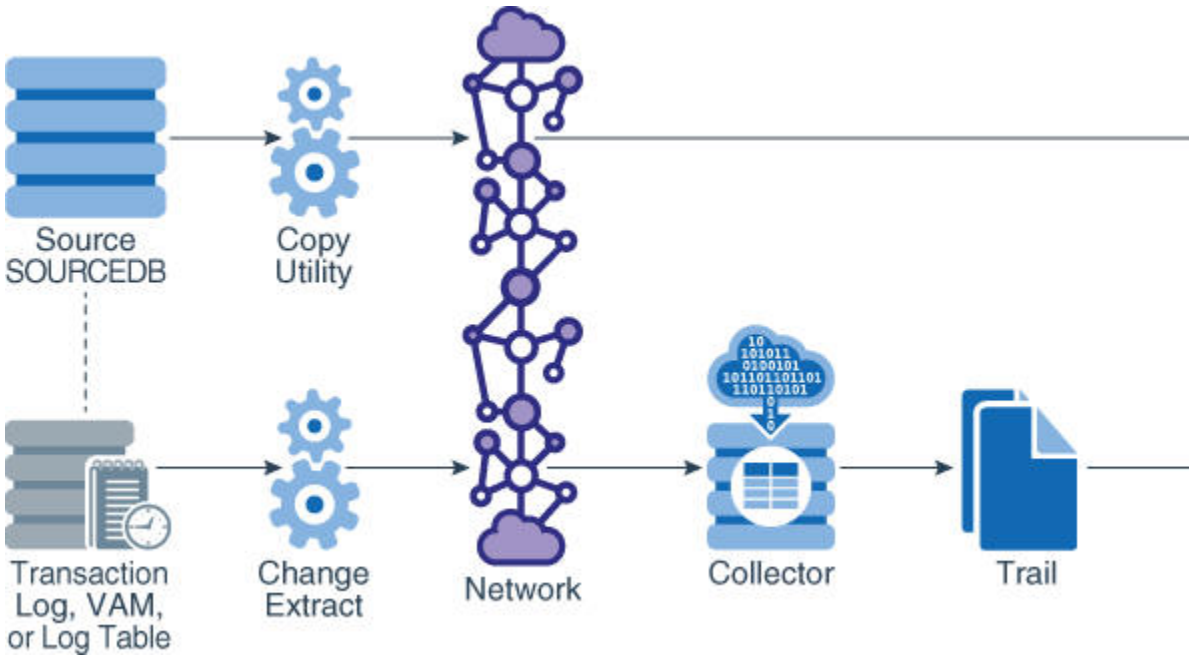
Table 7-1 Initial Load Replicat Parameters for Loading Data from File to Replicat

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.

7.5.2 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".

7.6 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

7.6.1 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

7.6.1.1 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.

7.6.1.2 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
```

7.6.2 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
```

7.6.3 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
```

7.6.4 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
```

7.6.5 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
```

7.7 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

7.7.1 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.

7.7.2 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.

7.8 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.

7.9 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.

7.10 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:

8.1 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.
```

8.2 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

8.2.1 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.

8.2.2 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 IV

Using Oracle GoldenGate with DB2 for z/OS

Oracle GoldenGate for DB2 for z/OS runs remotely on Linux, zLinux, or AIX. With Oracle GoldenGate, you can move data between similar or dissimilar supported DB2 for z/OS versions, or you can move data between a DB2 for z/OS database and a database of another type, such as Oracle or DB2 LUW. Oracle GoldenGate for DB2 for z/OS platform supports the filtering, mapping, and transformation of data.

This part describes tasks for configuring and running Oracle GoldenGate on a DB2 for z/OS database.

Topics:

9

Understanding What's Supported for DB2 for z/OS

This chapter contains support information for Oracle GoldenGate on DB2 for z/OS databases.

Topics:

9.1 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 GoldenGatee 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.

9.2 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

9.3 Supported Objects and Operations for DB2 for z/OS

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

- Extraction and replication of DML operations on DB2 for z/OS tables that contain rows of up to 512KB in length. This size exceeds the maximum row size of DB2.
- INSERT operations from the IBM LOAD utility are supported for change capture if the utility is run with LOG YES and SHRLEVEL CHANGE, and the source tables that are being loaded have DATA CAPTURE CHANGES enabled (required by Oracle GoldenGate) and are specified in the Oracle GoldenGate Extract configuration. Oracle GoldenGate also supports initial loads with the LOAD utility to instantiate target tables during initial synchronization. For more information, see *Administering Oracle GoldenGate*.
- 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).
- 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.

9.4 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:

10.1 Preparing Tables for Processing

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

10.1.1 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.

10.1.2 Assigning Row Identifiers

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.

- How Oracle GoldenGate Determines the Kind of Row Identifier to Use
- Using KEYCOLS to Specify a Custom Key

10.1.2.1 How Oracle GoldenGate Determines the Kind of Row Identifier to Use

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.

10.1.2.2 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*.

> **Note:**
>
> If you want to use the RRN of the records as the key for a table, you may access the `GGHEADER` Oracle GoldenGate Environment Variable `AUDITRBA` which will contain the RRN for each record processed.

10.1.3 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);
     ```

For more information about KEYCOLS, see Assigning Row Identifiers.

10.2 Configuring a Database Connection

This section contains instructions for setting up the Extract and Replicat connections to a SQL Server database.

- Setting Initialization Parameters
- Specifying the Path to the Initialization File
- Ensuring ODBC Connection Compatibility
- Specifying the Number of Connection Threads

10.2.1 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.`

10.2.2 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"
```

10.2.3 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.

10.2.4 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.

10.3 Accessing Load Modules

Grant Oracle GoldenGate USS access to the `SDSNLOAD` system load library and to the `DSNHDECP` load module. You can include the libraries in one of the following places:

- The z/OS system search order.
- The USS profile of the Oracle GoldenGate user. Use a UNIX command similar to the following, where `DSN810` is the user-assigned data set prefix from the DB2 installation.

```
export STEPLIB='DSN810.SDSNEXIT:DSN810.SDSNLOAD'
```

The preceding command will cause USS to allocate the equivalent of a `STEPLIB DD` statement whenever it executes a shell command or Oracle GoldenGate process. If using APF, all libraries in the `STEPLIB` concatenation must be APF-authorized.

10.4 Specifying Job Names and Owners

By default, USS sets the job name and owner of all Oracle GoldenGate processes to that of the user who started them. You can change the job name or user by setting the `_BPX_JOBNAME` and `_BPX_USERID` environment variables, or you can create z/OS jobs or started-task procedures for the Oracle GoldenGate processes. To use the environment variable `_BPX_JOBNAME`, at a minimum you should have read access to the RACF `FACILITY` class and `BPX.JOBNAME` name. For more details, see *Installing Oracle GoldenGate* and the IBM *z/OS System Services Planning* document.

10.5 Assigning WLM Velocity Goals

The user who starts the Manager process is typically the user by which other Oracle GoldenGate processes run. Oracle GoldenGate work appears as forked child processes of WLM subsystem type OMVS. Assign the Oracle GoldenGate processes their Workload Manager (WLM) velocity goals based on the following guidelines.

- Assign the Extract process that reads the transaction logs a medium velocity goal, one that is below the velocity of the main DB2 address spaces, but above the velocity of most online transactions, TSO/E sessions, and z/OS batch work. The higher the velocity goal, the more processor power that Extract will receive, and the less lag that it will experience.
- You can assign an initial-load Extract process a velocity goal, or you can treat it as a typical DB2 batch job. For more information about the initial-load processes, see *Administering Oracle GoldenGate*.
- You might need to assign the Replicat process a higher velocity goal. Although Replicat is a typical DB2 batch application, it might require more processing power to prevent backlogs and latency.
- You probably will run Oracle GoldenGate utilities, such as `DEFGEN` and LOGDUMP, only occasionally, so you can let them perform like the other UNIX terminal-oriented work.
- If executing stored procedures with the `SQLEXEC` command, make certain that they do not become a bottleneck for Oracle GoldenGate. Their priority should be close to that of the calling Extract or Replicat process. WLM executes them with that priority, but the z/OS system executes them under the priority of a stored procedure as defined by the DB2 and z/OS system programmers.
- If you run Oracle GoldenGate under the TSO/E `OMVS` command, the Oracle GoldenGate processes are subject to the system and WLM limits of the TSO/E user account, rather than those of the UNIX kernel. Very long TSO/E response times (up to 20 seconds), often with little service consumption, can be recorded for an OMVS user because of the way that OMVS polls for terminal input. This can affect those WLM goals that are based on response time.

You can use multiple WLM service classes for the Oracle GoldenGate processes. The following is an example of how to maintain relative priorities for Oracle GoldenGate and other work, from highest priority to the lowest:

1. z/OS system processes, including the UNIX kernel and IRLM.
2. DB2 for z/OS address spaces for the primary Extract group.
3. Primary Extract group configured for online or batch change synchronization, and any DB2 stored procedures that it calls.
4. z/OS transaction managers, such as CICS and IMS.
5. Collector (Server) for local Extract data pump, if used.
6. Local Extract data pump (reading from trail), if used.
7. Collector for remote trails (files received from a remote site). Such files include the QSAM file created with the Extract `RMTBATCH` parameter on a NonStop system.
8. Online Replicat groups and any DB2 stored procedures that they call.

9. Manager process (required only for startup of Oracle GoldenGate processes and trail cleanup).
10. GGSCI and other user UNIX and TSO/E terminal work.
11. Initial-load Extract and any DB2 stored procedures that it calls.
12. Initial-load Replicat and any DB2 stored procedures that it calls.
13. Other z/OS batch work.

10.6 Monitoring Processes

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

- Viewing Oracle GoldenGate Messages
- Identifying Oracle GoldenGate Processes
- Interpreting Statistics for Update Operations

10.6.1 Viewing Oracle GoldenGate Messages

If the system log process (`syslog` daemon `syslogd`) is running, USS routes Oracle GoldenGate messages to their configured destination by means of UNIX message priority. For more information about configuring `syslogd`, see the z/OS IP configuration documents and the *UNIX System Services Planning* document.

If `syslogd` is not running, Oracle GoldenGate writes its command output, status information, and error messages to the system console. You can redirect console messages to the Oracle GoldenGate USS session and to the Oracle GoldenGate report files by using the following UNIX command:

```
export _BPXK_JOBLOG=STDERR
```

10.6.2 Identifying Oracle GoldenGate Processes

The system management facility (SMF) typically creates a separate accounting record for each UNIX process, including Oracle GoldenGate processes. However, if a user invokes the UNIX shell by using the `OMVS` command with the default `SHAREAS` option, or if a user sets the environment variable `_BPX_SHAREAS` to `YES` , it could cause two or more processes to run in the same address space. SMF provides process identification only for the first process, but resource consumption is accumulated for all processes that are running. For Oracle GoldenGate, this means that the work probably will be recorded under the Manager process, which is named `mgr`.

If the DB2 accounting trace is also active to the SMF destination, DB2 will create an SMF accounting record for each of the following Oracle GoldenGate processes:

- Extract
- Replicat
- Manager, if performing maintenance on Oracle GoldenGate tables. Examples of Oracle GoldenGate tables are the marker table and the Replicat checkpoint table.
- GGSCI sessions that issue the Oracle GoldenGate `DBLOGIN` command to log into the database.

10.6.3 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.

10.7 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

10.7.1 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.

10.7.2 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:

11.1 Making Transaction Data Available

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.

11.1.1 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;
   ```

11.1.2 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.

11.1.3 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.

11.1.4 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.

11.1.5 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 V
Using Oracle GoldenGate with MySQL

Oracle GoldenGate for MySQL supports replication from a MySQL source database to a MySQL target database or to a supported database of another type to perform an initial load or change data replication.

This part describes tasks for configuring and running Oracle GoldenGate on a MySQL database.

Topics:

12 Understanding What's Supported for MySQL

This chapter contains support information for Oracle GoldenGate on MySQL databases.

Topics:

12.1 Character Sets in MySQL

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

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

Limitations of Support

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

  ```
  armscii8
  keybcs2
  utf16le
  geostd8
  ```

12.2 Supported MySQL Data Types

MySQL supports the following data types:

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

12.2.1 Limitations and Clarifications

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

- Oracle GoldenGate does not support BLOB or TEXT types when used as a primary key.

- Oracle GoldenGate supports UTF8 and UCS2 character sets. UTF8 data is converted to UTF16 by Oracle GoldenGate before writing it to the trail.
- UTF32 is not supported by Oracle GoldenGate.
- 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.
- To preserve transaction boundaries for a MySQL target, create or alter the target tables to the InnoDB transactional database engine instead of the MyISAM engine. MyISAM will cause Replicat records to be applied as they are received, which does not guarantee transaction integrity even with auto-commit turned off. You cannot roll back a transaction with MyISAM.
- 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`.

12.3 Supported Objects and Operations for MySQL

Oracle GoldenGate for MySQL supports to following objects and operations:

- Basic extraction and replication of DDL (data definition language) operations for MySQL 5.7.10 and later. Only the `CREATE TABLE`, `ALTER TABLE`, and `DROP TABLE` operations are supported.
- Oracle GoldenGate supports the extraction and replication of transactional tables.

- 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.
- 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 supports the following DML operations on source and target database transactional tables:
  - Insert operation
  - Update operation (compressed included)
  - Delete operation (compressed included); cascade delete queries result in the deletion of the child of the parent operation
  - Truncate operation
- Oracle GoldenGate can operate concurrently with MySQL native replication.
- Oracle GoldenGate supports the `DYNSQL` feature for MySQL.
- 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.
  - Heartbeat Table is not supported on MySQL 5.5.

12.4 Non-Supported MySQL Data Types

MySQL does not support the following data types:

- The `XML`, `SET`, and Geometry data types and similar are not supported.
- There is no support for the Interval data type.

13

Preparing and Configuring Your System for Oracle GoldenGate

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

Topics:

13.1 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 *Reference for Oracle GoldenGate*

13.2 Setting Logging Parameters

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

> **Note:**
>
> 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.

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

   To specify the index file path with `TRANLOGICOPTIONS` with `ALTLOGDEST`, use the following command format on Windows:

   ```
   TRANLOGOPTIONS ALTLOGDEST "C:\\Program Files\\MySQL\\logs\\binlog.index"
   ```

   On Linux, use this format:

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

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. This is required only if DDL replication is enabled.
   - `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.

- `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. Any other log format (`MIXED` or `STATEMENT`) causes Extract to abend.

> **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.

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.

13.3 Adding Host Names

Oracle GoldenGate gets the name of the database it is supposed to connect to from the `SOURCEDB` parameter. A successful connection depends on the localhost entry being properly configured in the system host file. To avoid issues that arise from improper local host configuration, you can use `SOURCEDB` in the following format:

`SOURCEDB` *`database_name@host_name`*

Where: *`database_name`* is the name of the MySQL instance, and *`host_name`* is the name or IP address of the local host. 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`.

13.4 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.

13.5 Preparing Tables for Processing

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

- Assigning Row Identifiers
- Limiting Row Changes in Tables That Do Not Have a Key
- Disabling Triggers and Cascade Constraints

13.5.1 Assigning Row Identifiers

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.

- How Oracle GoldenGate Determines the Kind of Row Identifier to Use
- Tables with a Primary Key Derived from a Unique Index
- How to Specify Your Own Key for Oracle GoldenGate to Use

13.5.1.1 How Oracle GoldenGate Determines the Kind of Row Identifier to Use

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.

13.5.1.2 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.

13.5.1.3 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.

13.5.2 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.

13.5.3 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`.

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.

13.6 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.

13.7 Configuring Bi-Directional Replication

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

1. Configure Oracle GoldenGate for high availability or active-active replication according to the instructions in the Overview of Replicat in *Administering Oracle GoldenGate*
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 *Administering 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 bi-directional replication (and likewise, will enhance recovery).

3. Edit the MySQL server configuration file to set the `auto_increment_increment` and `auto_increment_offset` parameters to avoid discrepancies that could be caused by the bi-directional operations. The following illustrates these parameters, assuming two servers: **ServerA** and **ServerB**.

   ServerA:

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

ServerB:

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

13.8 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`.

13.9 Establishing a Secure Database Connection to AWS Aurora MySQL

Oracle GoldenGate uses SSH tunneling to connect to the AWS Aurora instance using the .pem file and then runs the Oracle GoldenGate MySQL delivery from on-premise.

To set up SSH tunneling for a secure database connection to AWS Aurora MySQL, perform the following tasks:

Task1: Configuring SSH Tunneling

To configure SSH Tunneling:

```
ssh -i your_key.pem -v -f -L local port number:cluster end point:end
point port number ec2 host name  > mysql_remote_forward.log 2>&1
```

In the following example, note that the local port number is set to 3308 because the port number 3306 may be used by MySQL that's installed locally.

```
bash-4.1$ ssh -i "test.pem" -v -N -f -L
3308:test-cluster.cluster-copyxiqzdjjl.us-west-2.rds.amazonaws.com:3306
ec2-user@ec2-52-11-244-17.us-west-2.compute.amazonaws.com >
mysql_remote_forward.log 2>&1
```

If Oracle GoldenGate is running behind a proxy, then perform the following configuration:

```
ssh -i your_key.pem -v -f -L local port number:cluster end point:end point
port number ec2 host name -o command to bypass proxy in double quotes> >
mysql_remote_forward.log 2>&1
```

For example:

```
bash-4.1$ ssh -i "test.pem" -v -N -f -L 3308:test-cluster.cluster-copyxiqzdjjl.us-
west-2.rds.amazonaws.com:3306
```

```
ec2-user@ec2-52-11-244-17.us-west-2.compute.amazonaws.com -o "ProxyCommand=nc
-X connect -x www-proxy.us.oracle.com:80 %h %p" > mysql_remote_forward.log 2>&1
```

Task 2: Provide the Connection String in the Extract Parameter File

The remote user must have all the permissions. The following is used in both the source and target databases:

```
remotedb@127.0.0.1:3308  userid
remote-user, password
remote-user-password
```

Also, to prevent a connection timeout, add the following variable to ~/.ssh/config file, because AWS automatically disconnects connection after 60 secs of inactivity:

```
bash-4.1$ cat ~/.ssh/config
ServerAliveInterval 50
```

13.10 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.

Table 13-1 Other Parameters for Oracle GoldenGate for MySQL

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.

Table 13-1 (Cont.) Other Parameters for Oracle GoldenGate for MySQL

Parameter	Description

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).

13.11 Positioning Extract to a Specific Start Point

You can position the ADD EXTRACT and ALTER EXTRACT commands to a specific start point in the transaction logs with the following command.

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

- *group* is the name of the Oracle GoldenGate Extract group for which the start position is required.
- *log_num* is the log file number. For example, if the required log file name is test.000034, this value is 34. Extract will search for this log file.
- *log_pos* 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 log_pos as 4. The log_pos 0 or 1 are not valid offsets to start reading and processing.

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 and cannot exceed a length of 37. 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/.

Topics:

14.1 DDL Configuration Prerequisites and Considerations

The prerequisites for configuring DDL replication are as follows:

- DDL replication is supported for MySQL 5.7.10 and greater.
- Bidirectional filtering for DDL replication is not supported.
- Remote capture implementation 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 DDLs are ignored by the capture and you should create the heartbeat tables manually at the target.

14.2 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.

14.3 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.

14.4 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 abcor 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.

14.5 Troubleshooting DDL Replication

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.

14.6 Uninstalling 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.

Part VI
Using Oracle GoldenGate with SQL Server

With Oracle GoldenGate for SQL Server, you can capture transactional data from user tables of supported SQL Server versions and replicate the data to a SQL Server database or other supported Oracle GoldenGate targets, such as an Oracle Database or Big Data target.

Oracle GoldenGate for SQL Server supports data filtering, mapping, and transformations unless noted otherwise in this documentation. And beginning with Oracle GoldenGate 12.3, there will be two separate data capture methods. The first, which is referred to as Classic Capture, is the transaction log based capture method. The second method, newly introduced with Oracle GoldenGate 12.3, is the CDC Capture method. The Classic Extract binary is available at My Oracle Support, under Patches and Updates, and requires a Service Request in order to receive a password to download the binary. The CDC Extract binary is available on the Oracle Software Delivery Cloud.

This part describes tasks for configuring, and running Oracle GoldenGate on a SQL Server database.

- Understanding What's Supported for SQL Server
  This chapter contains the requirements for the system and database resources that support Oracle GoldenGate.
- Preparing the System for Oracle GoldenGate
- Preparing the Database for Oracle GoldenGate — Classic Capture
- Preparing the Database for Oracle GoldenGate — CDC Capture
  Process to configure database settings and supplemental logging to support CDC Capture.
- Requirements Summary for Classic Extract in Archived Log Only (ALO) Mode
- Requirements Summary for Capture and Delivery of Databases in an AlwaysOn Availability Group
  Oracle GoldenGate for SQL Server features Capture support of the Primary and read-only, synchronous mode Secondary databases of an AlwaysOn Availability Group, and Delivery to the Primary database.
- Oracle GoldenGate Classic Extract for SQL Server Standard Edition Capture
- CDC Capture Method Operational Considerations
  This section provides information about the SQL Server CDC Capture options, features, and recommended settings.

15 Understanding What's Supported for SQL Server

This chapter contains the requirements for the system and database resources that support Oracle GoldenGate.

Topics:

15.1 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))
  - (Classic Extract – varbinary (max)with FILETREAM)
- 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)
- From this version of Oracle GoldenGate 12.3 release, Oracle GoldenGate for SQL Server (CDC Extract only) can replicate column data that contains SPARSE settings.

> **Note:**
>
> The previous versions of Oracle GoldenGate 12.3 release didn't support column data with `SPARSE` settings.

- From this version of Oracle GoldenGate 12.3 release, the `FILESTREAM` feature is supported for CDC Extract. This feature is already available for Classic Extract.

> **Note:**
>
> For details on `SPARSE` and `FILESTREAM` support, see New Features - March 2018 in *Release Notes for Oracle GoldenGate*

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 multi byte character data types and multi byte data stored in character columns. Multi byte data is supported only in a like-to-like, SQL Server configuration. Transformation, filtering, and other types of manipulation are not supported for multi byte character data.
- If data capture the `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, extend the size. Use `sp_configure` to view the current value of `max text repl size` and adjust the option as needed.
- 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.
- From this release of Oracle GoldenGate 12.3, a `VARBINARY (MAX)` column with the `FILESTREAM` attribute is supported up to a size of 4 GB. Extract uses standard Win32 file functions to read the `FILESTREAM` file. Its supported for both Classic and CDC Extracts.

> **Note:**
>
> Previous versions of Oracle GoldenGate 12.3 does not support this feature for CDC Extracts.

- 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:

- (Classic Extract) Oracle GoldenGate supports tables with non-persisted computed columns, but does not capture change data for these columns, because the database does not write it to the transaction log. To replicate data for non-persisted computed columns, use the `FETCHCOLS` or `FETCHMODCOLS` option of the `TABLE` parameter to fetch the column data from the table. Keep in mind that there may be discrepancies caused by differences in data values between the time that the column was changed in the database and the time that Extract fetches the data for the transaction record that is being processed.
- (CDC Extract) 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.

15.2 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.

15.3 Supported Objects and Operations for SQL Server

The following objects and operations are supported:

- Oracle GoldenGate supports capture of transactional DML from user tables, and delivery to user tables and writeable views.
- (Classic Extract) Capture from tables of a SQL Server Standard Edition database requires a primary key on the tables. Tables for an Enterprise Edition database do not require a primary key.
- Oracle GoldenGate supports the capture and delivery of DML operations on tables that contain rows of up to 512 in length.
- `TEXT`, `NTEXT`, `IMAGE`, `VARBINARY`, `VARCHAR` (`MAX`), and `NVARCHAR` (`MAX`) columns are supported in their full size.
- Oracle GoldenGate supports the maximum sizes that are permitted for tables that are tracked by CDC (for Enterprise Edition) and Transactional Replication (for Standard Edition).
- Oracle GoldenGate supports capture from tables enabled with `PAGE` and `ROW` compression. For partitioned tables that use compression, all partitions must be enabled with compression.
- Oracle GoldenGate supports capture for partitioned tables if the table has the same physical layout across all partitions.

15.4 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 or Transactional Replication, such as `TRUNCATE` statements. Refer to SQL Server Books Online for a complete list of the operations that are limited by enabling SQL Server Change Data Capture (for Enterprise Edition) and Transactional Replication (for Standard Edition).
- Extraction or replication of DDL (data definition language) operations. To avoid any conflict due to DDL changes, it is recommended that you perform the following steps:

  1. Pause or Stop application data to the table or tables to be changed.

2. (CDC Extract) Ensure the CDC Capture job processes all remaining transactions.
3. Ensure Extract process all the transactions prior to making any DDL changes. An Event Marker table may help to ensure full completion.
4. Stop Extract.
5. At source, execute `DELETE TRANDATA` for the specific tables on which `ALTER TABLE` (DDL changes) statement has to be performed.
6. Execute `ALTER TABLE` statement to add or drop the column in or from the table.
7. (CDC Extract) For cases when all tables previously enabled with `TRANDATA` had `TRANDATA` removed, this will also disable CDC on the database and it will be necessary to reposition the Extract with `BEGIN NOW` prior to restarting it.
8. Re-enable `TRANDATA` for the same table(s) at source.
9. Start Extract.
10. Restart your application.

- Capture from views. The underlying tables can be extracted and replicated.
- 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 switching.
- (Classic Extract) Oracle GoldenGate does not support non-native SQL Server transaction log backups, such as those offered by third-party vendors. However, if using the `TRANLOGOPTIONS` parameter with the `ACTIVESECONDARYTRUNCATIONPOINT` option, Extract does not need to read from any transaction log backups, so any log backup utility may be used. For more information, see Preparing the Database for Oracle GoldenGate — Classic Capture.
- (CDC Extract) 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).
- (CDC Extract) Due to a limitation with SQL Server's Change Data Capture, `NOOPUPDATES` are not captured by the SQL Server CDC agent so there are no records for Extract to capture for no-op update operations.

16

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:

16.1 Configuring a Database Connection

This section contains instructions for setting up the Extract and Replicat connections to a SQL Server database.

16.1.1 Configuring an Extract Database Connection

Extract connects to a source SQL Server database through an ODBC (Open Database Connectivity) connection. To create this connection, you set up a data source name (DSN) through the Data Sources (ODBC) control panel. See Configuring an ODBC Connection for instructions.

When using Transport Layer Security (TLS) 1.2 in your database environment, include the following parameter in Extract to force it to use a supported connection protocol:

```
DBOPTIONS DRIVER SQLNCLI11
```

16.1.2 Configuring a Replicat Database Connection

Replicat can connect to the target database to perform DML operations in the following ways:

- Through ODBC.
- Through OLE DB. This is the default and provides slightly better performance than using ODBC.
- Through OLE DB as the SQL Server replication user. `NOT FOR REPLICATION` must be set on `IDENTITY` columns, foreign key constraints, and triggers.

> **Note:**
>
> Because Replicat always uses ODBC to query for metadata, you must configure a target ODBC connection.

Before you select a method to use, review the following guidelines and procedures to evaluate the advantages and disadvantages of each.

- Using ODBC or Default OLE DB
- Using OLE DB with USEREPLICATIONUSER

16.1.2.1 Using ODBC or Default OLE DB

If Replicat connects through ODBC or through the default OLE DB connection, the following limitations apply:

- To keep `IDENTITY` columns identical on source and target when using ODBC or default OLE DB, Replicat creates special operations in its transaction to ensure that the seeds are incremented on the target. These steps may reduce delivery performance.
- You must adjust or disable triggers and constraints on the target tables to eliminate the potential for redundant operations.

To use Replicat with either ODBC or OLE DB, follow these steps:

1. To use ODBC exclusively, include the `DBOPTIONS` parameter with the `USEODBC` option in the Replicat parameter file. (To use the default OLE DB connection, no parameter is required.)
2. Disable triggers and constraints on the target tables. See Disabling Triggers and Cascade Constraints.
3. To use `IDENTITY` columns in a bidirectional SQL Server 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 two-server installation would be as follows:
   - Sys1 sets the seed value at 1 with an increment of 2.
   - Sys2 sets the seed value at 2 with an increment of 2.

   A three-server installation would be as follows:
   - Sys1 sets the seed value at 1 with an increment of 3.
   - Sys2 sets the seed value at 2 with an increment of 3.
   - Sys3 sets the seed value at 3 with an increment of 3.
4. Configure an ODBC data source. See Configuring an ODBC Connection.

> **Note:**
>
> OLE DB uses the ODBC connection settings to derive connection information for OLE DB together with information on which driver to use.

16.1.2.2 Using OLE DB with USEREPLICATIONUSER

If Replicat connects as the SQL Server replication user through OLE DB with the USEREPLICATIONUSER option, and NOT FOR REPLICATION is enabled for IDENTITY, triggers, and foreign key constraints, the following benefits and limitations apply.

- IDENTITY seeds are not incremented when Replicat performs an insert. For SQL Server bidirectional configurations, stagger the seed and increment values like the example in Step 3 of the previous section.
- Triggers are disabled for the Replicat user automatically on the target to prevent redundant operations. However triggers fire on the target for other users.
- Foreign key constraints are not enforced on the target for Replicat transactions. CASCADE updates and deletes are not performed. These, too, prevent redundant operations.
- CHECK constraints are not enforced on the target for Replicat transactions. Even though these constraints are enforced on the source before data is captured, consider whether their absence on the target could cause data integrity issues.

> **Note:**
>
> Normal IDENTITY, trigger, and constraint functionality remains in effect for any users other than the Replicat replication user.

To use Replicat with USEREPLICATIONUSER, follow these steps:

1. In SQL Server Management Studio (or other interface) set the NOT FOR REPLICATION flag on the following objects. For active-passive configurations, set it only on the passive database. For active-active configurations, set it on both databases.
   - Foreign key constraints
   - Check constraints
   - IDENTITY columns
   - Triggers (requires textual changes to the definition; see the SQL Server documentation for more information.)
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. 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.

4. In the Replicat parameter file, include the `DBOPTIONS` parameter with the `USEREPLICATIONUSER` option.
5. Configure an ODBC data source. See Configuring an ODBC Connection.

16.1.3 Configuring an ODBC Connection

Follow these instructions to create a SQL Server system data source name (DSN) for a source SQL Server database and for a target SQL Server database. A DSN stores information about how to connect to a SQL Server database through ODBC (Open Database Connectivity).

> **Note:**
>
> Even when using OLE DB as the apply connection method, Replicat always uses ODBC to query the target database for metadata. Therefore Replicat always requires a DSN.

To create a SQL Server DSN

1. To run the ODBC client, select **Control Panel**, select **Administrative Tools**, and then select **Data Sources (ODBC)**.
2. In the ODBC Data Source Administrator dialog box of the ODBC client, select the System DSN tab, and then click **Add**.
3. Under Create New Data Source, select the correct SQL Server driver for the edition, and then click **Finish**. The Create a New Data Source to SQL Server wizard appears.
4. Supply the following, and then 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**: Select the SQL Server instance name.
5. For login authentication, do one of the following, and then click **Next**:
   a. Select **With Integrated Windows Authentication** for Oracle GoldenGate to use Windows authentication.
   b. To use database credentials, select **With SQL Server authentication using a login ID and password entered by the user**, and supply the login information.
6. If the default database is not set to the one that Oracle GoldenGate will connect to, click **Change the default database to**, and then select the correct name. Set the other settings to use ANSI. 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 on each SQL Server source and target system.

16.2 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
- Assigning Row Identifiers
- Improving IDENTITY Replication with Array Processing

16.2.1 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.

16.2.2 Assigning Row Identifiers

Oracle GoldenGate requires unique row identifiers on the source and target tables to locate the correct target rows for replicated updates and deletes. Source tables can have any kind of key listed in How Oracle GoldenGate Determines the Kind of Row Identifier to Use, except for tables of a SQL Server Standard Edition instance, which require a primary key. If there is no primary key identified on a table that has fixed-length columns, the length of one of the fixed-length columns must be below 3800 bytes.

- How Oracle GoldenGate Determines the Kind of Row Identifier to Use

- Using KEYCOLS to Specify a Custom Key

16.2.2.1 How Oracle GoldenGate Determines the Kind of Row Identifier to Use

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 (required for tables of a Standard Edition instance).
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. If neither of these key types exist , Oracle GoldenGate constructs a pseudokey 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 SQL Server, Oracle GoldenGate requires the row data in target tables that do not have a primary key to be less than 8000 bytes.

> **Note:**
>
> If there are types of keys on a table or if there are no keys at all on a table, Oracle GoldenGate logs a 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.

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

16.2.3 Improving `IDENTITY` Replication with Array Processing

Because only one table per session can have `IDENTITY_INSERT` set to `ON`, Replicat must continuously toggle `IDENTITY_INSERT` when it applies `IDENTITY` data to multiple tables in a session. To improve the performance of Replicat in this situation, use the `BATCHSQL` parameter. `BATCHSQL` causes Replicat to use array processing instead of applying SQL statements one at a time.

16.3 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. For more information, see *Administering Oracle GoldenGate*.

17

Preparing the Database for Oracle GoldenGate — Classic Capture

This section contains information that helps you configure database settings and supplemental logging to support Classic Capture of source transaction data by Oracle GoldenGate.

Topics:

17.1 Setting the Database to Full Recovery Model

Oracle GoldenGate requires that you set a SQL Server source database to the Full recovery model and that a full database backup exists.

To verify or set the recovery model:

1. Connect to the SQL Server instance with SQL Server Management Studio for SQL Server.
2. Expand the Databases folder.
3. Right-click the source database, and then select **Properties**.
4. Select the **Options** tab.
5. Under Recovery, set the model to **Full**, if it is not already set.
6. If the database was in simple recovery or never had a full database backup, take a full database backup before positioning Extract.
7. Click **OK**.

17.2 Backing Up the Transaction Log

Oracle GoldenGate Capture for SQL Server can run in two modes: real-time capture mode and archived log only mode.

In archived log only mode, Extract reads only from transaction log backups.

In real-time capture mode, the Extract process may occasionally require information from a log backup depending on how you configure the management of the secondary truncation point. This requirement occurs when log records have been flushed to log backups and are no longer available in the online log.

In either mode, Oracle GoldenGate Capture for SQL Server requires that the log backup files meet the following conditions:

- The log backup is a native SQL Server log backup made by issuing the `BACKUP LOG` command (or the corresponding GUI command). Third-party log backups are not supported.
- The log backup can be compressed using native SQL Server compression features.
- The log backup is made to a DISK device. Valid examples include:

```
BACKUP LOG dbname TO DISK = "c:\folder\logbackup.trn"
BACKUP LOG dbname TO DISK = "\\server\share\logbackup.trn"
```

Additional recommendations:

- Do not overwrite existing log backups.
- Striped log backups are not supported.
- Appending log backups to the same file is not recommended.
- Mixing compressed and uncompressed log backups to the same device or file is not supported.

17.3 Enabling Supplemental Logging

These instructions apply to source installations of Oracle GoldenGate for all supported versions of SQL Server .

When you enable supplemental logging with the `ADD TRANDATA` command, Extract can capture the information that is required to reconstruct transactions. By default, SQL Server logs only some of this information.

`ADD TRANDATA` must be issued for all tables that are to be captured by Oracle GoldenGate. `ADD TRANDATA` uses different features of SQL Server, depending on the edition of SQL Server.

- In Enterprise Edition, Change Data Capture (CDC) for SQL Server creates a minimal Change Data Capture object on the specified table.
- In Standard Edition, a SQL Server transactional publication is created, as well as an article for each table that is enabled with `TRANDATA`

  .

Enterprise Edition

- Enables the database for SQL Server Change Data Capture by running `EXECUTE sys.sp_cdc_enable_db`.
- Creates a Change Data Capture table for each base table enabled with supplemental logging by running `EXECUTE sys.sp_cdc_enable_table`.
- Oracle GoldenGate does not use CDC tables except as necessary to enable supplemental logging.
- When SQL Server enables CDC, SQL Server creates two jobs per database:
  - `cdc.dbname_capture`
  - `cdc.dbname_cleanup`

- Make sure to review the information on TRANLOGOPTIONS options in Managing the Secondary Truncation Point for a complete understanding of this required parameter and the interaction it has with the secondary truncation point and these SQL Server jobs.

Standard Edition

- To enable TRANDATA for tables of a SQL Server Standard Edition database, SQL Server Replication features must be installed, a Distributor must be configured, and a distribution database must be created.
- An Oracle GoldenGate transactional publication is created for the database, and an article for each table is created in the database's publication. The articles are set to not replicate the data to a distribution database. However, they are needed to enable supplemental logging for SQL Server Standard Edition.
- When a transactional publication is created, a SQL Server Log Reader Agent job is created for the database. The Log Reader Agent job has a naming convention of *instancename-dbname-#* and job category of REPL-LogReader.
- Make sure to review the information on TRANLOGOPTIONS options in Managing the Secondary Truncation Point for a complete understanding of this required parameter and the interaction it has with the secondary truncation point and these SQL Server jobs.
- To Enable Supplemental Logging

17.3.1 To Enable Supplemental Logging

This procedure requires 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 this command:

- SOURCEDB *DSN* is the name of the SQL Server data source.
- USERID *user* is the database login and PASSWORD *password* that is required if the data source connects via SQL Server authentication. If the credentials are stored in a credential store, USERIDALIAS *alias* is the alias for the credentials. If you are 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. 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.*
```

17.4 Managing the Secondary Truncation Point

When you enable supplemental logging with the `ADD TRANDATA` command for at least one table in a source SQL Server database, a secondary truncation point is created in the transaction log that has to be moved for log space to be released as needed, following subsequent log backups. Use the `TRANLOGOPTIONS` parameter to control how the secondary truncation point is managed. `TRANLOGOPTIONS` is a required parameter for a SQL Server Extract and has these available options for managing the secondary truncation point, one of which must be chosen:

- Oracle GoldenGate Manages the Secondary Truncation Point
- SQL Server Manages the Secondary Truncation Point

17.4.1 Oracle GoldenGate Manages the Secondary Truncation Point

Two `TRANLOGOPTIONS` options instruct the Oracle GoldenGate Extract process to manage the secondary truncation point: `ACTIVESECONDARYTRUNCATIONPOINT` and `MANAGESECONDARYTRUNCATIONPOINT`. You can use either of these options if you have the following situations:

- Extract *is not* running concurrently (for the same source database) with SQL Server transactional replication.
- SQL Server Change Data Capture (CDC) that is configured by applications other than Oracle GoldenGate.
-

> **Note:**
>
> Using `TRANLOGOPTIONS ACTIVESECONDARYTRUNCATIONPOINT` or `MANAGESECONDARYTRUNCATIONPOINT` for Extract when either SQL Server transactional replication and/or CDC configured by applications other than Oracle GoldenGate are running at the same time causes the SQL Server Log Reader Agent or CDC capture job to fail.

The `TRANLOGOPTIONS`option and the `ACTIVESECONDARYTRUNCATIONPOINT`option has different characteristics that are described in this section.

The `ACTIVESECONDARYTRUNCATIONPOINT` option has the following characteristics:

- Extract sets the secondary truncation point in the transaction log only for those transactions that it has captured.
- `ACTIVESECONDARYTRUNCATIONPOINT` prevents the truncation of the transactions until Extract has captured them from the online transaction log.
- Because no online data can be truncated before Extract captures it, Extract is never required to read from log backups. The advantage of this method is that you can use third-party transaction log backup software without concern that Extract may need to access the log backups to retrieve unprocessed transaction data.

- Using option `ACTIVESECONDARYTRUNCATIONPOINT` may result in larger log files if Extract has significant lag. Therefore, you may need to adjust your storage parameters to accommodate the larger logs. Larger logs occur because transaction log backups cannot immediately truncate the transaction log. Truncation occurs only after Extract has finished capturing the data.
- Using this option requires that you allow only one Extract to capture from the source database.
- This option is not valid for an Extract that is running in archived log mode.
- If you are using SQL Server Enterprise Edition, when the Extract starts, it stops and deletes the CDC capture job for the database, preventing data from being loaded to any CDC tables. If any transactions for the tables occurs between the time that supplemental logging is added and Extract starts, the corresponding CDC tables may have data in them. This data is not used by Oracle GoldenGate and can be manually deleted or truncated from the CDC tables. Additionally, the CDC cleanup job for the source database can be disabled or deleted. It no longer functions once the CDC capture job has been removed.
- If you are using SQL Server Standard Edition, you must manually stop and disable the SQL Server Log Reader Agent job for the source database, before you position the Extract.

The `MANAGESECONDARYTRUNCATIONPOINT` option has the following characteristics:

- Extract commonly sets the truncation point in the transaction log at preset interval, by setting a high water mark for the truncation point at that point in the log.
- As subsequent transaction log backups are made, transactions before the truncation point can be removed from the online transaction log, which frees space in the online log. If Extract has not captured a transaction from the online log before it is flushed to a log backup, it switches to the log backup and reads the transaction from the backup. This ability requires the logs to be backed up only with a native SQL Server backup utility. Third-party backup software cannot be used in this scenario.
- Using this option lets multiple Extracts capture from the same source database. Configure Extract with this option if there are multiple Extracts for a single source database.
- If you are using SQL Server Enterprise Edition starts, it stop and deletes the CDC capture job for the database, preventing data from being loaded to any CDC tables. If any transactions for the tables occurs between the time that supplemental logging is added and Extract starts, the corresponding CDC tables may have data in them. This data is not used by Oracle GoldenGate and can be manually deleted or truncated from the CDC tables. Additionally, the CDC cleanup job for the source database can be disabled or deleted. It no longer functions once the CDC capture job has been removed.
- If you are using SQL Server Standard Edition, you must manually stop and disable the SQL Server Log Reader Agent job for the source database.

When you use either of these options, if Extract is stopped or down for a longer period of time than the log backup frequency, the secondary truncation point does not move and transactions are not removed from the transaction log after subsequent log backups. This may cause the log file to grow to its maximum size. If this occurs, contact Oracle Support for information about manually moving the secondary truncation point.

17.4.2 SQL Server Manages the Secondary Truncation Point

Use `TRANLOGOPTIONS` with the `NOMANAGESECONDARYTRUNCATIONPOINT` option if Extract will run concurrently (for the same source database) with SQL Server transactional replication and/or SQL Server CDC that is configured by applications other than Oracle GoldenGate. SQL Server will manage the secondary truncation point.

17.5 Retaining the Log Backups and Backup History

Retain enough log backups and backup history (in the msdb database) 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 data in the transaction log or a log backup that contains the start of the oldest uncommitted unit of work, and all log backups thereafter.

If data that Extract needs during processing is not retained, either in online logs or in the backups, 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.

To determine where the Extract checkpoints are, use the `INFO EXTRACT SHOWCH` command. For more information, see Replicat Checkpoints in *Oracle Fusion Middleware Administering Oracle GoldenGate for Windows and UNIX Guide*.

18

Preparing the Database for Oracle GoldenGate — CDC Capture

Process to configure database settings and supplemental logging to support CDC Capture.

This section contains information that helps you configure database settings and supplemental logging to support CDC Capture of source transaction data by Oracle GoldenGate.

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:

18.1 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 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'
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.*
```

18.2 Retaining the CDC Table History 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 operations within a transaction. And as part of enabling supplemental logging, SQL Server will create its own Change Data Capture Cleanup job that runs nightly by default, and purges data older than 72 hours. The SQL Server CDC Cleanup job is unaware that an Extracts 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 this situation, Oracle GoldenGate for CDC Extract includes a .bat binary that will create an Oracle GoldenGate Cleanup job and associated stored procedures and tables. Each CDC Extract, upon startup, will expect, by default that those Oracle GoldenGate Cleanup job objects exist, and the Extract will shut down if they do not. The Extract will also shutdown if the SQL Server CDC Cleanup job exists alongside the Oracle GoldenGate Cleanup job. The default checks by Extract for the Oracle GoldenGate CDC Cleanup objects can be overwritten by using the `TRANLOGOPTIONS NOMANAGECDCCLEANUP` in the Extract, but this would only be recommended in development and testing situations.

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 re-enable this feature should any of the objects get manually deleted.

To create the Oracle GoldenGate CDC Cleanup job and objects:

This requires an 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 binary

1. 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.bat` file, providing the following variable values:

   ```
   ogg_cdc_cleanup_setup.bat createJob userid password databasename servername
   \instancename schema
   ```

   In the preceding, *`userid password`* is a valid SQL Server login and password for the user with the `sysadmin` rights. The source database name and instance name are *`databasename servername\instancename`*. 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`

The Oracle GoldenGate CDC Cleanup job when created, is scheduled to run every ten minutes, with a default retention period of seventy two hours. 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.

18.3 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
   ```

19

Requirements Summary for Classic Extract in Archived Log Only (ALO) Mode

Oracle GoldenGate for SQL Server includes a feature of capturing DML from only the SQL Server transaction log backups GoldenGate can run on the database server in an ALO configuration, or optionally, GoldenGate can be installed and run on a middle tier Windows server. It should be pointed out that when using an ALO mode configuration, replication will have an induced lag which will be based on the log backup interval as well as the time it takes to complete writing out each log backup during that interval, and the time that it takes the Extract to fully process the log backup file.

Topics:

19.1 Windows OS Requirements

- Optional requirements if installing and running GoldenGate from a middle tier Windows server:
  - Install SQL Server Client Tools on the middle tier server in order to configure primary database ODBC connection
    - Microsoft SQL Server 2008 SP4 Feature Pack: `https://www.microsoft.com/en-us/download/details.aspx?id=44277`
    - Microsoft SQL Server 2008 R2 SP3 Feature Pack: `https://www.microsoft.com/en-us/download/details.aspx?id=44272`
    - Microsoft SQL Server 2012 Feature Pack: `https://www.microsoft.com/en-us/download/details.aspx?id=29065`
    - Microsoft SQL Server 2014 Feature Pack: `https://www.microsoft.com/en-us/download/details.aspx?id=42295`
  - Set the middle tier server's date, time, and time zone to the same as the primary source database server.
  - Create a network share of the folder that contains the source database transaction log backups. For example, if SQL Server writes log backups to `D:\SQLBackups`, then create a share on this folder that can be accessed by the Extract running on the middle tier Windows server.
- Oracle GoldenGate Manager must run as an account with READ permissions to the log backup folder, the log backups, and the network share if configuring for remote ALO mode capture.

  - The default of Local System Account will work if 'Everybody' has share and folder access (not very secure).
  - Oracle recommends that you use a Windows account to run the Manager service and control share and folder access to that account.

19.2 Transaction Log Backups

- Set up SQL Server transaction log backups at a tolerable interval, every 15 minutes for example, if this is still within the recovery SLA for the database.
  - The log backup frequency will be minimum data lag for replication as well.
- Set the transaction log backups to write to the folder that is shared on the network if installing GoldenGate on a middle tier Windows server.

19.3 ODBC Connection

- For an Extract in ALO mode running on the database server, create a System DSN for the database as per normal instructions.
- For an Extract in ALO mode running on a middle tier Windows server, create a system DSN on that server that connects back to the primary database. This connection will be used by Extract to retrieve table metadata, manage the secondary truncation point, and other tasks.

19.4 Supplemental Logging

Supplemental logging must be enabled by normal means (`ADD TRANDATA`) via `GGSCI` against the source database tables to be included for capture.

> **Note:**
>
> Tables to be captured only from the ALO mode are still required to have Supplemental Logging enabled.

19.5 Operational Requirements and Considerations

- Extract must use either `TRANLOGOPTIONS MANAGESECONDARYTRUNCATIONPOINT` or `NOMANAGESECONDARYTRUNCATIONPOINT`.

  `ACTIVESECONDARYTRUNCATIONPOINT` is not compatible with the ALO mode and will cause Extract to Abend if it is enabled.
- Extract must also use the `TRANLOGOPTIONS ARCHIVEDLOGONLY` parameter, which instructs the Extract to capture DML from the transaction log backups only.
- For remote ALO capture, use `TRANLOGOPTIONS ALTARCHIVELOGDEST` as well, listing the path of the network share location (in double quotes) where the SQL Server transaction log backups are located. Extract will query the `msdb` database to find the full path of the SQL Server log backup that is needed, and substitute the value of the path from the query result with the path listed in `ALTARCHIVELOGDEST`.

20

Requirements Summary for Capture and Delivery of Databases in an AlwaysOn Availability Group

Oracle GoldenGate for SQL Server features Capture support of the Primary and read-only, synchronous mode Secondary databases of an AlwaysOn Availability Group, and Delivery to the Primary database.

Topics:

20.1 ODBC Connection

- Create a System DSN that connects to the Primary (Capture/Delivery) or Secondary (Capture only) replica database as per normal instructions.
- If the connection used in the DSN is to the AG Listener, then Capture will only be against the Primary database.

20.2 Supplemental Logging

Supplemental logging must be enabled by normal means (`ADD TRANDATA`) using GGSCI against the Primary database tables to be included for capture, and not against a Secondary replica database.

- If Oracle GoldenGate is installed on a secondary replica node, you will need to create a separate DSN to the Primary database, in order to connect using `DBLOGIN` and run `ADD TRANDATA`.
- The login used with `DBLOGIN` must have `sysadmin` membership of the Primary replica instance.
- When enabling supplemental logging on the Primary database, the SQL Server Change Data Capture job does not automatically get created, therefore, if a Secondary database becomes a Primary after failover, then you must manually create the SQL Server CDC capture job on the new Primary by running the following against the new Primary:

  ```
  EXECUTE sys.sp_cdc_add_job 'capture'
  ```

  – If you modify the job's parameters from its default values using `EXEC sys.sp_cdc_change_job`, then after adding the job to the new Primary database, they would also have to re-run `EXEC sys.sp_cdc_change_job` against the capture job on the new Primary database.

> **Note:**
>
> Consult the Microsoft documentation on how to enable the CDC Capture job for AlwaysOn Secondary Replicas for more information:.

20.3 Operational Requirements and Considerations

- (CDC Extract) When running an Extract from a middle tier Windows server, set the middle tier server's date, time, and time zone to the same as that of the Primary database server.
- (CDC Extract) If an Extract is reading from a readable Secondary database, then it will not be able to update the Oracle GoldenGate CDC Cleanup job if used, therefore, Extract must run with `TRANLOGOPTIONS NOMANAGECDCCLEANUP` when running against a secondary database.
- (CDC Extract) If an Extract is configured to capture against the Primary database, upon failover to a new Primary, manually create theOracle GoldenGate CDC Cleanup job before starting the Extract. This only needs to be done once on each potential Primary after it is available. It can be done by scripting the job from the previous Primary and running against the new Primary.

21

Oracle GoldenGate Classic Extract for SQL Server Standard Edition Capture

Classic Extract for Oracle GoldenGate for SQL Server is designed to capture DML from both SQL Server Standard Edition and SQL Server Enterprise Edition.
Topics:

21.1 Overview

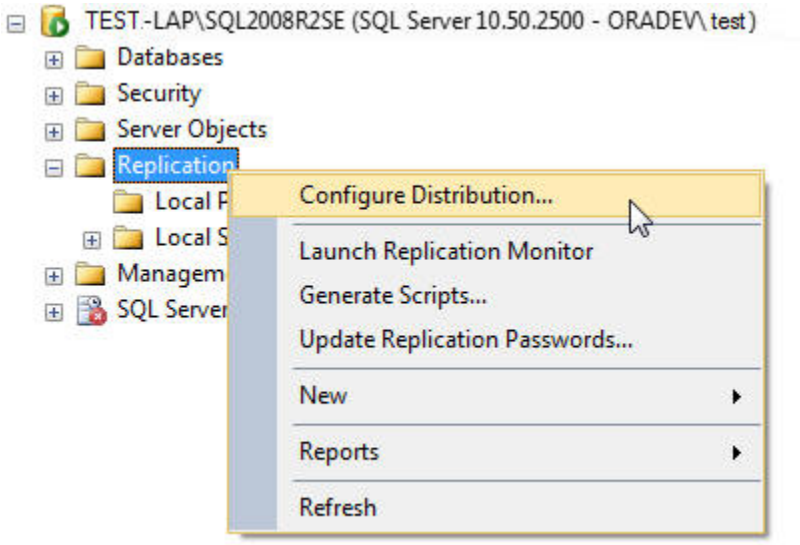
Oracle GoldenGate for SQL Server includes a Classic Capture support for SQL Server Standard Edition. Oracle GoldenGate utilizes certain SQL Server Replication components in order to enable supplemental logging. These SQL Server Replication components are required to be installed and configured in order to enable supplemental logging, and the instructions and limitations are outlined in the following sections.

21.2 SQL Server Instance Requirements

SQL Server Replication components must be installed. This is normally done with the initial installation of SQL Server. If its not already installed, you can re-run the SQL Server installer and add that feature to the existing instance. To install SQL Server Replication components, perform the following steps:

1. On the SQL Server Installation Center screen, select the **New installation or add features to an existing installation** option.
2. On the Installation Type page, select the **Add features to an existing instance of SQL Server** option.
3. Select the developer features to install on the feature selection page.

Once the SQL Server Replication components are installed, a Distributor must be configured, along with a distribution database. These steps must be done manually prior to attempting to enable Supplemental Logging for any table in the database.



The Distributor can be a local or a remote Distributor, and can be one that has already been configured for an existing SQL Server Replication implementation. Oracle GoldenGate does not require the distribution database to store change data, but it must be configured in order to enable supplemental logging.

21.3 Table Requirements

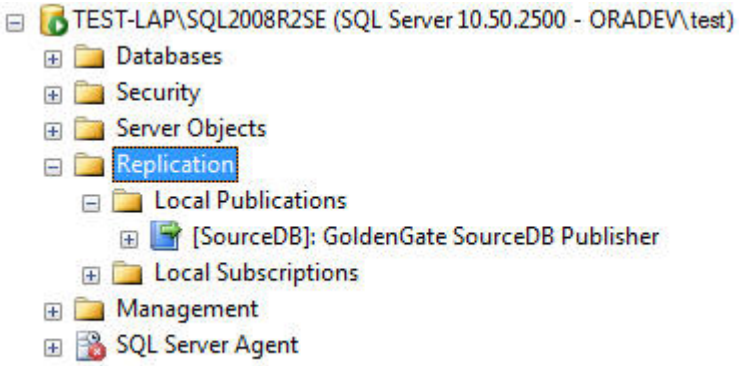
When enabling supplemental logging for tables within SQL Server Standard Edition, the tables must contain a Primary Key.

21.4 Supplemental Logging

When enabling supplemental logging for tables within SQL Server Standard Edition, a new SQL Server Replication Publication is created, and the tables enabled for supplemental logging will be added to the Publication as Articles.

If there are no previous Publications for the database, a new Log Reader job will be created under SQL Server Agent, and depending on the secondary truncation point management method used by Extract, the job may need to be manually stopped and disabled. The created Publication has the naming convention of: "[*DatabaseName*]: GoldenGate *DatabaseName* Publisher".

Here is an example of how to manually stop the SQL Server reader job.



The Article properties of the tables to be configured with supplemental logging will not log data changes to the distribution database, but the creation of the Publication with Articles is the requirement to enabling supplemental logging.

21.5 Operational Requirements and Considerations

- If Oracle GoldenGate Capture will not be used in conjunction with other SQL Server Transactional Publications for the source database, it is recommended to let the Extract manage the secondary truncation point by one of the two available options listed in See Reference..
- Allowing Oracle GoldenGate to manage the secondary truncation point requires that you manually stop and disable the database Log Reader job SQL Server Agent.
- The database Log Reader job will have a similar naming convention as `Server \Instance-DatabaseName-1`and will be of the job category `REPL-LogReader` when its properties are viewed.
- To stop and disable the job, right-click the job name under **SQL Server Agent within Management Studio** then select **Stop Job**, follow the prompts to stop the job, then right-click again, and then click **Disable**.

22 CDC Capture Method Operational Considerations

This section provides information about the SQL Server CDC Capture options, features, and recommended settings.

22.1 Tuning the SQL Server Change Data Capture Job

The SQL Server Change Data Capture job collects data from the SQL Server transaction log and loads it into the CDC 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'
```

22.2 Valid and Invalid 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 fpr Windows and UNIX*.

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. Therefore secondary truncation point management is not handled by the Extract, and for the Change Data Capture Extract, these parameters are not valid.

22.3 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`, and 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.

There are other options to the utility as well, and those will be discussed below.

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 binary.

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. Ensure that the Extract has been deleted via `GGSCI: DELETE EXTRACT <extname>`

2. On the source system, open a command prompt and change to the Oracle GoldenGate installation folder.

3. Run the `ogg_cdc_cleanup_setup.bat` file, providing the following variable values:

```
ogg_cdc_cleanup_setup.bat deleteExtCheckpoint <userid> <password> <databasename>
<servername\instancename> <schema>
```

Example: `ogg_cdc_cleanup_setup.bat deleteExtCheckoint ggsuser ggspword db1 server1\inst1 ogg`

4. You will be prompted to enter the name of the Extract that is to be removed, and once entered, press the Enter/Return key to continue.

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`

22.4 Changing from Classic Extract to a CDC Extract

If you plan to change from using a Classic Extract from Oracle GoldenGate 12.3 or earlier, to a 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, and the information provide 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.

Part VII
Using Oracle GoldenGate with Teradata

Only Oracle GoldenGate release 12*c* (12.3.0.1) and later for Teradata support the delivery of data from other types of databases to a Teradata database.

This part describes the tasks for configuring and running Oracle GoldenGate on a Teradata database.

- Overview of Oracle GoldenGate for Teradata
  Oracle GoldenGate for Teradata supports the filtering, mapping, and transformation of data unless noted otherwise in this documentation.
- Understanding What's Supported for Teradata
  This chapter contains support information for Oracle GoldenGate on Teradata databases.
- Preparing the System for Oracle GoldenGate
- Configuring Oracle GoldenGate
- Common Maintenance Tasks

Overview of Oracle GoldenGate for Teradata

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

High-speed Oracle GoldenGate replication can be used to refresh a Teradata cache environment with minimal latency. In addition, with its heterogeneous support, Oracle GoldenGate enables the Teradata data store to be used as a data integration point for other data sources.

23

Understanding What's Supported for Teradata

This chapter contains support information for Oracle GoldenGate on Teradata databases.

Topics:

23.1 Supported Teradata Data Types

Table 23-1 shows the Teradata data types that Oracle GoldenGate supports. Any limitations or conditions that apply follow this table.

Table 23-1 Supported Data Types by Oracle GoldenGate, Per Teradata Version

Data type	v12	v13	v13.1
BLOB	No	Yes	Yes
BYTEINT	Yes	Yes	Yes
VARBYTE	Yes	Yes	Yes
BIGINT	Yes	Yes	Yes
BYTEINT	Yes	Yes	Yes
DATE	Yes	Yes	Yes
DECIMAL - 18 and under	Yes	Yes	Yes
DECIMAL - 19 to 38	No	Yes	Yes
DOUBLE PRECISION	Yes	Yes	Yes
FLOAT	Yes	Yes	Yes
INTEGER	Yes	Yes	Yes
NUMERIC - 18 and under	Yes	Yes	Yes
NUMERIC - 19 to 38	No	Yes	Yes
REAL	Yes	Yes	Yes
SMALLIINT	Yes	Yes	Yes
TIME	Yes	Yes	Yes
TIMESTAMP	Yes	Yes	Yes
INTERVAL	Yes	Yes	Yes

Table 23-1 (Cont.) Supported Data Types by Oracle GoldenGate, Per Teradata Version

Data type	v12	v13	v13.1
INTERVAL DAY	Yes	Yes	Yes
INTERVAL DAY TO HOUR	Yes	Yes	Yes
INTERVAL DAY TO MINUTE	Yes	Yes	Yes
INTERVAL DAY TO SECOND	Yes	Yes	Yes
INTERVAL HOUR	Yes	Yes	Yes
INTERVAL HOUR TO MINUTE	Yes	Yes	Yes
INTERVAL HOUR TO SECOND	Yes	Yes	Yes
INTERVAL MINUTE	Yes	Yes	Yes
INTERVAL MINUTE TO SECOND	Yes	Yes	Yes
INTERVAL MONTH	Yes	Yes	Yes
INTERVAL SECOND	Yes	Yes	Yes
INTERVAL YEAR	Yes	Yes	Yes
INTERVAL YEAR TO MONTH	Yes	Yes	Yes
CHAR	Yes	Yes	Yes
CLOB	No	Yes	Yes
CHAR VARYING	Yes	Yes	Yes
LONG VARCHAR	Yes	Yes	Yes
VARCHAR	Yes	Yes	Yes
GRAPHIC	Yes	Yes	Yes
LONG VARGRAPHIC	Yes	Yes	Yes
VARGRAPHIC	Yes	Yes	Yes
PERIOD (DATE)	No	Yes	Yes
PERIOD (TIME)	No	Yes	Yes
PERIOD (TIMESTAMP)	No	Yes	Yes
UDT	No	Yes	Yes

- Limitations of Support for Numeric Data Types
- Limitations of Support for Single-byte Character Data Types
- Conditions and Limitations of Support for Multi-byte Character Data
- Limitations of Support for Binary Data Types
- Limitations of Support for Large Object Data Types

23.1.1 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.

23.1.2 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.

23.1.3 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.

23.1.4 Limitations of Support for Binary Data Types

No limitations. These data types are supported between other source databases and Teradata targets.

23.1.5 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.

23.1.6 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.

23.1.7 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.

23.2 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:
  - Teradata does not have any internal event/job schedulers so automatic updating and inserting of records into Heartbeat tables *cannot* occur.
  - 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 tables *cannot* occur. As such, you should explicitly drop or truncate the corresponding heartbeat objects to suit your environment.

23.3 Non-Supported Operations for Teradata

This section lists the data operations that Oracle GoldenGate does not support.

- Extract (capture)

- DDL

24

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:

24.1 Preparing Tables for Processing

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

24.1.1 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.

24.1.2 Assigning Row Identifiers

Oracle GoldenGate requires unique row identifiers on the source and target tables to locate the correct target rows for replicated updates and deletes. Source tables can have any kind of key listed in How Oracle GoldenGate Determines the Kind of Row Identifier to Use, except for tables of a SQL Server Standard Edition instance, which require a primary key. If there is no primary key identified on a table that has fixed-length columns, the length of one of the fixed-length columns must be below 3800 bytes.

- Using KEYCOLS to Specify a Custom Key

24.1.2.1 How Oracle GoldenGate Determines the Kind of Row Identifier to Use

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 (required for tables of a Standard Edition instance).
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. If neither of these key types exist , Oracle GoldenGate constructs a pseudokey 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 SQL Server, Oracle GoldenGate requires the row data in target tables that do not have a primary key to be less than 8000 bytes.

> **Note:**
>
> If there are types of keys on a table or if there are no keys at all on a table, Oracle GoldenGate logs a 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.

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

25
Configuring Oracle GoldenGate

This chapter describes how to configure Oracle GoldenGate Replicat. This chapter contains the following sections:
Topics:

25.1 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
   ```

   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 25-1 plus any others that apply to your database environment.

Example 25-1 Parameters for the Replicat Group

```
-- Identify the Replicat group:
REPLICAT rep
-- State whether or not source and target definitions are identical:
SOURCEDEFS {full_pathname | ASSUMETARGETDEFS}
-- Specify database login information as needed for the database:
[TARGETDB dsn2,] [USERID user id[, PASSWORD pw]]
-- Specify error handling rules (See the NOTE following parameter file):
REPERROR (error, response)
-- Specify tables for delivery:
MAP owner.table, TARGET owner.table[, DEF template name];
```

> **Note:**
>
> In a recovery situation, it is possible that Replicat could attempt to apply some updates twice. If a multiset table is affected, this could result in duplicate rows being created. Use the `REPERROR` parameter in the Replicat parameter file so that Replicat ignores duplicate rows.

25.2 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

25.2.1 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;
```

25.2.2 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.

25.2.3 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*.

26
Common Maintenance Tasks

This chapter contains instructions for performing some common maintenance tasks when using the Oracle GoldenGate replication solution.
Topics:

26.1 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.