Oracle® Fusion Middleware

Installing and Upgrading Oracle GoldenGate for Big Data

12*c* (12.3.2.1)
F11081-01
October 2018

ORACLE

Oracle Fusion Middleware Installing and Upgrading Oracle GoldenGate for Big Data, 12*c* (12.3.2.1)

F11081-01

Primary Author: Lorna Vallad

Contents

2 Upgrading Oracle GoldenGate for Big Data

Preface

Understanding Oracle GoldenGate12*c* (12.3.0.1) describes data replication concepts, Oracle GoldenGate Classic Architecture, Oracle GoldenGate Microservices Architecture, and the architecture components. Using these concepts you can implement a data replication solution using Oracle GoldenGate.

Audience

This guide is intended for system administrators who are configuring and running Oracle GoldenGate for Big Data.

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`.

Accessible Access to Oracle Support

Oracle customers who 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.

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.

Related Information

The Oracle GoldenGate Product Documentation Libraries are found at

Oracle GoldenGate

Oracle GoldenGate Application Adapters

Oracle GoldenGate for Big Data

Oracle GoldenGate Plug-in for EMCC

Oracle GoldenGate Monitor

Oracle GoldenGate for HP NonStop (Guardian)

Oracle GoldenGate Veridata

Oracle GoldenGate Studio

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

Oracle GoldenGate A-Team Chronicles

1

Installing Oracle GoldenGate for Big Data

This chapter describes how to install a new instance of Oracle GoldenGate for Big Data.

Topics:

Understanding What's Supported

Oracle GoldenGate for Big Data supports specific configurations: the handlers, which are compatible with clearly defined software versions, and there are many support topics. This section provides the relevant support information.

Topics:

Verifying Certification and System Requirements

Make sure that you install your product on a supported hardware or software configuration. For more information, see the certification document for your release on the *Oracle Fusion Middleware Supported System Configurations* page.

Oracle has tested and verified the performance of your product on all certified systems and environments; whenever new certifications occur, they are added to the proper certification document right away. New certifications can occur at any time, and for this reason the certification documents are kept outside of the documentation libraries on the Oracle Technology Network.

The compatibility of the Oracle GoldenGate for Big Data Handlers with the various data collections, including distributions, database releases, and drivers is included in the certification document.

Understanding Handler Compatibility

This section describes the compatibility of the Oracle GoldenGate for Big Data Handlers with the various data collections, including distributions, database releases, and drivers.

Topics:

Cassandra Handler

The Cassandra Handler uses the Datastax Java Driver for Apache Cassandra. This driver streams change data capture from a source trail file into the corresponding tables in the Cassandra database.

The Cassandra Handler is designed to work with the following versions :

Distribution	Version
Apache Cassandra	1.2 2.0 2.1 2.2 3.*x* 3.3.1
Datastax Enterprise Cassandra	3.2 4.0 4.5 4.6 4.7 4.8

Note:

You must upgrade the Cassandra driver to version 3.3.1 if you are using a different version.

Flume Handler

The Oracle GoldenGate for Big Data Flume Handler works with the Apache Flume versions 1.6.*x*, 1.5.*x* , 1.4.*x*, and 2.8.0. Compatibility with versions of Flume before 1.4.0 is not guaranteed.

The Flume Handler is compatible with the following versions:

Distribution	Version
Distribution: Apache Flume	Version: 1.7.*x*, 1.6.*x*, 1.5.*x*, and 1.4.*x*
Hortonworks Data Platform (HDP)	HDP 2.6 (Flume 1.5.2) HDP 2.5 (Flume 1.5.2) HDP 2.4 (Flume 1.5.2) HDP 2.3 (Flume 1.5.2) HDP 2.2 (Flume 1.5.2) HDP 2.1 (Flume 1.4.0)
Cloudera Distribution Including Apache Hadoop (CDH)	CDH 5.11*x* (Flume 1.6.0) CDH 5.10*x* (Flume 1.6.0) CDH 5.9.*x* (Flume 1.6.0) CDH 5.8.*x* (Flume 1.6.0) CDH 5.7.*x* (Flume 1.6.0) CDH 5.6.*x* (Flume 1.6.0) CDH 5.5.*x* (Flume 1.6.0) CDH 5.4.*x* (Flume 1.5.0) CDH 5.3.*x* (Flume 1.5.0) CDH 5.2.*x* (Flume 1.5.0) CDH 5.1.*x* (Flume 1.5.0)

Elasticsearch Handler

The Elasticsearch Handler is designed to work with the following versions :

Distribution	Version
Elasticsearch 2.X	- 2.0.X - 2.1.X - 2.2.X - 2.3.X - 2.4.X
Elasticsearch 5.X	- 5.0.X - 5.1.X - 5.2.X

HBase Handler

Cloudera HBase 5.4.*x* and later did not fully adopt the Apache HBase 1.0.0 client interface so it is not fully in sync with the Apache HBase code line to provide reverse compatibility in that HBase client interface. This means that Cloudera HBase broke

binary compatibility with the new HBase 1.0.0 interface resulting in `NoSuchMethodError` when integrating with the Oracle GoldenGate for Big Data HBase Handler. This can be solved one of the following two ways:

- Configure the HBase Handler to use the 0.98.*x* HBase interface by setting the HBase Handler configuration property, `hBase98Compatible`, to `true`.
- Alternatively, you can use the Apache HBase client libraries when connecting to CDH 5.4.*x* and later HBase.

The Cloudera HBase compatibility issue is solved by dynamically using Java Reflection. You can use Oracle GoldenGate 12.3.1.1.0 or later to stream data to HBase in Cloudera CDH 5.4.*x* and later, You simply leave the `hBase98Campatibility` property unset or set to `false`, which uses the newer HBase 1.0 interface.

The HBase Handler is designed to work with the following:

Distribution	Version
Apache HBase	0.98.*x* and 0.96.*x* when you set the `hBase98Compatible` property to true 1.2.*x*, 1.1.*x* and 1.0.*x*
Hortonworks Data Platform (HDP)	HDP 2.6 (HBase 1.1.2) HDP 2.5 (HBase 1.1.2) HDP 2.4 (HBase 1.1.2) HDP 2.3 (HBase 1.1.1) HDP 2.2 (HBase 0.98.4) when you set the `hBase98Compatible` property to `true`.
Cloudera Distribution Including Apache Hadoop (CDH)	CDH 5.11.*x* (HBase 1.2.0) CDH 5.10.*x* (HBase 1.2.0) CDH 5.9.*x* (HBase 1.2.0) CDH 5.8.*x* (HBase 1.2.0) . CDH 5.7.*x* (HBase 1.2.0). CDH 5.6.*x* (HBase 1.0.0). CDH 5.5.*x* (HBase 1.0.0). CDH 5.4.*x* (HBase 1.0.0). CDH 5.3.*x* (HBase 0.98.6) when you set the `hBase98Compatible` property to `true`. CDH 5.2.*x* (HBase 0.98.6) when you set the `hBase98Compatible` property to `true`. CDH 5.1.*x* (HBase 9.98.1) when you set the `hBase98Compatible` property to `true`.

HDFS Handler

The HDFS Handler is designed to work with the following versions :

Distribution	Version
Apache Hadoop	2.8.*x* 2.7.*x* 2.6.0 2.5.*x* 2.4.*x* 2.3.0 2.2.0 3.0.0-alpha
Hortonworks Data Platform (HDP)	HDP 2.6 (HDFS 2.7.3) HDP 2.5 (HDFS 2.7.3) HDP 2.4 (HDFS 2.7.1) HDP 2.3 (HDFS 2.7.1) HDP 2.2 (HDFS 2.6.0) HDP 2.1 (HDFS 2.4.0)
Cloudera Distribution Include Apache Hadoop (CDH)	CDH 5.11.*x* (HDFS 2.6.0) CDH 5.10.*x* (HDFS 2.6.0) CDH 5.9.*x* (HDFS 2.6.0) CDH 5.8.*x* (HDFS 2.6.0) CDH 5.7.*x* (HDFS 2.6.0) CDH 5.6.*x* (HDFS 2.6.0) CDH 5.5.*x* (HDFS 2.6.0) CDH 5.4.*x* (HDFS 2.6.0) CDH 5.3.*x* (HDFS 2.5.0) CDH 5.2.*x* (HDFS 2.5.0) CDH 5.1.*x* (HDFS 2.3.0)

JBDC Handler

The JDBC handler internally uses generic JDBC API. Although it should be compliant with any JDBC complaint database driver we have certified the JDBC handler against the following targets:

- Oracle Database target using Oracle JDBC driver.
- MySQL Database target using MySQL JDBC driver.
- IBM Netezza target using Netezza JDBC driver.
- Amazon Redshift target using Redshift JDBC driver.
- Greenplum target using the Oracle branded DataDirect Greenplum JDBC driver. Contact Oracle support for the branded JDBC driver files.

Kafka and Kafka Connect Handlers

These handlers are *not* compatible with Kafka version 8.2.2.2 and later.

These handlers are designed to work with the following:

Distribution	Version
Apache Kafka	0.11.0.*x* 0.10.2.*x* 0.10.1.*x* 0.10.0.*x* 0.9.0.*x*
Hortonworks Data Platform (HDP)	HDP 2.6 (Kafka 0.10.1.2) HDP 2.5 (Kafka 0.10.0) HDP 2.4 (Kafka 0.9.0)
Cloudera Distribution Including Apache Hadoop (CDH) does not currently include Kafka. Cloudera currently distributes Kafka separately as Cloudera Distribution of Apache Kafka	Cloudera Distribution of Apache Kafka 2.1.*x* (Kafka 0.10.0.0) Cloudera Distribution of Apache Kafka 2.0.*x* (Kafka 0.9.0.0)
Confluent Platform	3.2.*x* (Kafka 0.10.2.*x*) 3.1.*x* (Kafka 0.10.1.*x*) 3.0.*x* (Kafka 0.10.0.*x*) 2.0.0 (Kafka 0.9.0.0)

Kinesis Streams Handler

The Kinesis Streams Handler is hosted on the Amazon cloud so does not have a public version. The assumption is that compatibility to Kinesis is assured as long as compatibility to the Amazon Software Development Kit (SDK) is maintained. The Kinesis Streams Handler was developed using the 1.11.*x* version of the Amazon SDK.

MongoDB Handler

The MongoDB handler uses the native Java driver version 3.2.2. It is compatible with the following MongoDB versions:

- MongoDB 2.4
- MongoDB 2.6
- MongoDB 3.0
- MongoDB 3.2
- MongoDB 3.4

What are the Additional Support Considerations?

This section describes additional Oracle GoldenGate for Big Data Handlers additional support considerations.

Pluggable Formatters—Support
The handlers support the Pluggable Formatters as described in Using the Pluggable Formatters as follows:

- The HDFS Handler supports all of the pluggable handlers .

- Pluggable formatters are not applicable to the HBase Handler. Data is streamed to HBase using the proprietary HBase client interface.
- The Flume Handler supports all of the pluggable handlers described in .
- The Kafka Handler supports all of the pluggable handlers described in .
- The Kafka Connect Handler does *not* support pluggable formatters. You can convert data to JSON or Avro using Kafka Connect data converters.
- The Kinesis Streams Handler supports all of the pluggable handlers described in .
- The Cassandra, MongoDB, and JDBC Handlers do *not* use a pluggable formatter.

Avro Formatter—Improved Support for Binary Source Data
In previous releases, the Avro Formatter did not support the Avro bytes data type. Binary data was instead converted to Base64 and persisted in Avro messages as a field with a string data type. This required an additional conversion step to convert the data from Base64 back to binary.
The Avro Formatter now can identify binary source fields that will be mapped into an Avro bytes field and the original byte stream from the source trail file will be propagated to the corresponding Avro messages without conversion to Base64.

Avro Formatter—Generic Wrapper
The `schema_hash` field was changed to the `schema_fingerprint` field. The `schema_fingerprint` is a long and is generated using the `parsingFingerprint64(Schemas)` method on the `org.apache.avro.SchemaNormalization` class. This identifier provides better traceability from the Generic Wrapper Message back to the Avro schema that is used to generate the Avro payload message contained in the Generic Wrapper Message.

JSON Formatter—Row Modeled Data
The JSON formatter supports row modeled data in addition to operation modeled data.. Row modeled data includes the after image data for insert operations, the after image data for update operations, the before image data for delete operations, and special handling for primary key updates.

Java Delivery Using Extract
Java Delivery using Extract is *not* supported and was deprecated in this release. Support for Java Delivery is only supported using the Replicat process. Replicat provides better performance, better support for checkpointing, and better control of transaction grouping.

Kafka Handler—Versions
Support for Kafka versions 0.8.2.2, 0.8.2.1, and 0.8.2.0 was discontinued. This allowed the implementation of the flush call on the Kafka producer, which provides better support for flow control and checkpointing.

HDFS Handler—File Creation
A new feature was added to the HDFS Handler so that you can use Extract, Load, Transform (ELT). The new `gg.handler.name.openNextFileAtRoll=true` property was added to create new files immediately when the previous file is closed. The new file appears in the HDFS directory immediately after the previous file stream is closed. This feature does not work when writing HDFS files in Avro Object Container File (OCF) format or sequence file format.

MongoDB Handler—Support

- The handler can only replicate unique rows from source table. If a source table has no primary key defined and has duplicate rows, replicating the duplicate rows to the MongoDB target results in a duplicate key error and the Replicat process abends.
- Missed updates and deletes are undetected so are ignored.
- Untested with sharded collections.
- Only supports date and time data types with millisecond precision. These values from a trail with microseconds or nanoseconds precision are truncated to millisecond precision.
- The `datetime` data type with `timezone` in the trail is not supported.
- A maximum BSON document size of 16 MB. If the trail record size exceeds this limit, the handler cannot replicate the record.
- No DDL propagation.
- No truncate operation.

JDBC Handler—Support

- The JDBC handler uses the generic JDBC API, which means any target database with a JDBC driver implementation should be able to use this handler. There are a myriad of different databases that support the JDBC API and Oracle cannot certify the JDBC Handler for all targets. Oracle has certified the JDBC Handler for the following RDBMS targets:

  Oracle
  MySQL
  Netezza
  Redshift
  Greenplum

- The handler supports Replicat using the `REPERROR` and `HANDLECOLLISIONS` parameters, see *Reference for Oracle GoldenGate*.
- The database metadata retrieved through the Redshift JDBC driver has known constraints, see *Release Notes for Oracle GoldenGate for Big Data*.

  Redshift target table names in the Replicat parameter file must be in lower case and double quoted. For example:

  ```
  MAP SourceSchema.SourceTable, target “public”.”targetable”;
  ```

- DDL operations are ignored by default and are logged with a `WARN` level.
- Coordinated Replicat is a multithreaded process that applies transactions in parallel instead of serially. Each thread handles all of the filtering, mapping, conversion, SQL construction, and error handling for its assigned workload. A coordinator thread coordinates transactions across threads to account for dependencies. It ensures that DML is applied in a synchronized manner preventing certain DMLs from occurring on the same object at the same time due to row locking, block locking, or table locking issues based on database specific rules. If there are database locking issue, then Coordinated Replicat performance can be extremely slow or pauses.

Delimited Formatter—Limitation
Handlers configured to generate delimited formatter output only allows single character delimiter fields. If your delimiter field length is greater than one character, then the handler displays an error message similar to the following and Replicat abends.

```
oracle.goldengate.util.ConfigException: Delimiter length cannot be more than one
character. Found delimiter [|||]
```

DDL Event Handling
Only the `TRUNCATE TABLE` DDL statement is supported. All other DDL statements are ignored.
You can use the `TRUNCATE` statements one of these ways:

- In a DDL statement, `TRUNCATE TABLE`, `ALTER TABLE TRUNCATE PARTITION`, and other DDL `TRUNCATE` statements. This uses the `DDL` parameter.
- Standalone `TRUNCATE` support, which just has `TRUNCATE TABLE`. This uses the `GETTRUNCATES` parameter.

Preparing for Installation

Prepare your Java environment by ensuring that you have the correct version of Java installed, and that the environmental variables have been set up and configured correctly.

- Installation Overview
- Downloading Oracle GoldenGate for Big Data
- Installing Java
- Directory Structure
- Setting up Environmental Variables

Installation Overview

This section provides an overview of the installation contents and the Oracle GoldenGate instances used with the Oracle GoldenGate Adapter

- Contents of the Installation ZIP File
- Using the Generic Build of Oracle GoldenGate
- Considerations for Using a Custom Build for a Big Data Instance of Oracle GoldenGate
- Installing to a Non-Generic Instance of Oracle GoldenGate

Contents of the Installation ZIP File

The Oracle GoldenGate for Big Data installation ZIP file contains:

- Oracle GoldenGate Java Adapter
- A version of Oracle GoldenGate designed to stream data to Big Data targets. This version is labeled *generic* because it is not specific to any database, but it is platform dependent.

Using the Generic Build of Oracle GoldenGate

For JMS capture, the Java Adapter must run in the generic build of Oracle GoldenGate. However, the generic build is not required when using the adapter for delivery of trail data to a target; in this case, the Java Adapter can be used with any database version of Oracle GoldenGate.

Considerations for Using a Custom Build for a Big Data Instance of Oracle GoldenGate

There are both advantages and disadvantages to installing a custom build for a Big Data Oracle GoldenGate instance. Also, there are limitations in the releases of Oracle GoldenGate that are compatible with releases of the Big Data.

Advantages

- The non-generic instance allows you to configure Extract to login to the database for metadata. This removes the need to use a source definitions file that must be synchronized your the source database DDL.
- There is no need to manage two separate versions of Oracle GoldenGate when doing database capture and JMS delivery on the same server.

Disadvantages

- If you need to patch Oracle GoldenGate core instance, you must also copy the Big Data into the new patched installation of Oracle GoldenGate.
- The Oracle GoldenGate for Big Data are only tested and certified with the generic version of Oracle GoldenGate core. New patches of the core can trigger incompatibilities.

Limitations

- The Replicat module to write to Big Data targets is only available in the Generic Oracle GoldenGate distribution.
- The Oracle GoldenGate for Big Data can be installed with the same major release as your Oracle GoldenGate instance. Therefore, 11.1.*x* releases of the Big Data can only be installed to 11.1.x releases of Oracle GoldenGate; 11.2.*x* with 11.2.*x*, and 12.1.2.*x* with 12.1.2.*x*.
- The Oracle GoldenGate for Big Data release 12.3.0.*x* reads Oracle GoldenGate release 12.2.*x* and earlier trail files. The Oracle GoldenGate for Big Data release 12.3.1.*x* reads Oracle GoldenGate release 12.3.*x* and earlier trail files.
- The generic build must be used with JMS capture, as this is the only version of Extract that is capable of loading the VAM.
- A `DEFGEN` utility is not included with the Big Data. To generate source definitions, you will need a version of Oracle GoldenGate that is built specifically for your database type.

Installing to a Non-Generic Instance of Oracle GoldenGate

If you decide to install the Java user exit to a non-generic instance of Oracle GoldenGate, unzip to a temporary location first and then copy the adapter files to your Oracle GoldenGate installation location

To do this, follow these steps:

1. Extract the Oracle GoldenGate installation ZIP file to a temporary directory.
2. Extract the Oracle GoldenGate installation ZIP file into your Oracle GoldenGate installation directory.
3. Copy or move the files from the temporary directory `ggjava` subdirectory into the Oracle GoldenGate installation directory.
4. Copy or move the shared libraries from the temporary location into the Oracle GoldenGate installation directory.
5. Optionally you can also copy `Gendef`. (There is no need to copy the shared library `ggjava_vam` because it only works with the generic build.)
6. Delete the temporary directory.

Downloading Oracle GoldenGate for Big Data

Oracle GoldenGate for Big Data are available for Windows, Linux, and UNIX. To download, first visit the Oracle support site to see if there is a patch available for your operating system and architecture.

> **Note:**
>
> If you are not planning to use the generic build included in the installation, ensure that the major release of the Oracle GoldenGate for Big Data build you download matches (or is known to be compatible with) the major release of the Oracle GoldenGate instance that will be used with it.

1. Navigate to `http://support.oracle.com`.
2. Sign in with your Oracle ID and password.
3. Select the Patches and Upgrades tab.
4. On the Search tab, click Product or Family.
5. In the Product Field, type **Oracle GoldenGate for Big Data**.
6. From the Release drop-down list, select the release version that you want to download.
7. Make sure Platform is displayed as the default in the next field, and then select the platform from the drop-down list.
8. Leave the last field blank.
9. Click **Search**.

10. In the Advanced Patch Search Results list, select the available builds that satisfy the criteria that you supplied.
11. In the file Download dialog box, click the ZIP file to begin the download.

If patches are not available on the support site, go to the Oracle delivery site for the release download.

1. Navigate to `http://edelivery.oracle.com`.
2. Sign in with your Oracle ID and password.
3. On the Terms and Restrictions page:
   - Accept the **Trial License Agreement** (even if you have a permanent license).
   - Accept the **Export Restrictions**.
   - Click **Continue**.
4. On the Media Pack Search page:
   - Select the Oracle Fusion Middleware Product Pack.
   - Select the platform on which you will be installing the software.
   - Click **Go**.
5. In the Results list:
   - Select the Oracle GoldenGate Applications Big Data Media Pack that you want.
   - Click **Continue**.
6. On the Download page:
   - View the `Readme` file.
   - Click **Download** for each component that you want. Follow the automatic download process to transfer the zip file to your system.

Installing Java

The Oracle GoldenGate for Big Data are certified for Java 1.8. Before installing and running Oracle GoldenGate for Java, you must install Java (JDK or JRE) version 1.8 or later. Either the Java Runtime Environment (JRE) or the full Java Development Kit (which includes the JRE) may be used.

Directory Structure

The following table is a sample that includes the subdirectories and files that result from unzipping the installation file and creating the subdirectories. The following conventions have been used:

- Subdirectories are enclosed in square brackets []
- Levels are indicated by a pipe and hyphen |-
- The `Internal` notation indicates a read-only directory that should not be modified
- Text files (`*.txt`) are not included in the list
- Oracle GoldenGate utilities, such as Defgen, Logdump, and Keygen, are not included in the list

Table 1-1 Sample installation directory structure

Directory	Explanation
[gg_install_dir]	Oracle GoldenGate installation directory, such as C:/ggs on Windows or /home/user/ggs on UNIX.
|-ggsci	Command line interface used to start, stop, and manage processes.
|-mgr	Manager process.
|-extract	Extract process that will start the Java application.
|-replicat	Replicat process that will start the Java application.
|-[UserExitExamples]	Sample C programming language user exit code examples.
|-[dirprm]	Subdirectory that holds all the parameter and property files created by the user, for example: javaue.prm javaue.properties jmsvam.prm jmsvam.properties ffwriter.prm
|-[dirdef]	Subdirectory that holds source definitions files (*.def) defining the metadata of the trail: • Created by the core utility for the user exit trail data. • Created by the Gendef adapter utility for VAM message capture.
|-[dirdat]	Subdirectory that holds the trail files produced by the VAM Extract or read by the user exit Extract.
|-[dirrpt]	Subdirectory that holds log and report files.
|-[dirchk]	Internal Subdirectory that holds checkpoint files.
|-[dirpcs]	InternalSubdirectory that holds process status files.
|-[dirjar]	Internal Subdirectory that holds Oracle GoldenGate Monitor jar files.
|-[ggjava]	Internal Installation directory for Java jars. Read-only; do not modify.
|-|-ggjava.jar	The main Java application jar that defines the class path and dependencies.
|-|-[resources]	Subdirectory that contains all ggjava.jar dependencies. Includes subdirectories for: • [class] - properties and resources • [lib] - application jars required by ggjava.jar
|-ggjava_vam.dll	The VAM shared library. This is libggjava_vam.so on UNIX.

Table 1-1 (Cont.) Sample installation directory structure

Directory	Explanation
`|-ggjava.dll`	Used by the Replicat based delivery process. This is `libggjava.so` on UNIX.
`|-gendef`	Utility to generate the adapter source definitions files containing metadata of the JMS message input (useful only for trail files created by Oracle GoldenGate releases 12.1 or older. Note that this is different from the Oracle GoldenGate `Defgen` utility that creates source definitions containing the input metadata for the trail.
`|-. . .`	Other subdirectories and files included in the installation or created later.

Setting up Environmental Variables

To configure your Java environment for Oracle GoldenGate for Java:

- The `PATH` environmental variable should be configured to find your Java Runtime
- The shared (dynamically linked) Java virtual machine (JVM) library must also be found.

On Windows, these environmental variables should be set as system variables; on Linux/UNIX, they should be set globally or for the user running the Oracle GoldenGate processes. Examples of setting these environmental variables for Windows, UNIX, and Linux are in the following sections.

> **Note:**
>
> There may be two versions of the `JAVA_HOME/.../client`, and another in `JAVA_HOME/.../server`. For improved performance, use the server version, if it is available. On Windows, only the client JVM may be there if only the JRE was installed (and not the JDK).

- Java on Linux/UNIX
- Java on Windows

Java on Linux/UNIX

Configure the environment to find the JRE in the `PATH`, and the JVM shared library, using the appropriate environmental variable for your system. For example, on Linux (and Solaris, etc.), set `LD_LIBRARY_PATH` to include the directory containing the JVM shared library as follows (for `sh/ksh/bash`):

> **Note:**
>
> On AIX platforms, you set `LIBPATH=`. On HP-UX IA64, you set `SHLIB_PATH=`.

Example 1-1 Configuring path for Java on Linux

```
export JAVA_HOME=/opt/jdk1.8
export PATH=$JAVA_HOME/bin:$PATH
export LD_LIBRARY_PATH=$JAVA_HOME/jre/lib/i386/server:$LD_LIBRARY_PATH
```

In the examples above, the directory `$JAVA_HOME/jre/lib/i386/server` should contain the `libjvm.so` and `libjsig.so` files. The actual directory containing the JVM library depends on the operating system and if the 64-bit JVM is being used.

Verify the environment settings by opening a command prompt and checking the Java version as in this example:

```
$ java -version
java version "1.8.0_92"
Java(TM) SE Runtime Environment (build 1.8.0_92-b14)
```

Java on Windows

After Java is installed, configure the `PATH` to find the JRE and JVM DLL (`jvm.dll`):

Example 1-2 Configuring Path for Java on Windows

```
set JAVA_HOME=C:\Program Files\Java\jdk1.8.0
set PATH=%JAVA_HOME%\bin;%PATH%
set PATH=%JAVA_HOME%\jre\bin\server;%PATH%
```

In the example above, the directory `%JAVA_HOME%\jre\bin\server` should contain the file `jvm.dll`.

Verify the environment settings by opening a command prompt and checking the Java version as in this example:

```
C:\> java -version
java version "1.8.0_92" Java(TM) SE Runtime Environment (build 1.8.0_92-b14))
```

Installation Steps

Perform the following steps to install the Oracle GoldenGate for Big Data:

1. Create an installation directory that has no spaces in its name. Then extract the ZIP file into this new installation directory. For example:

   ```
   Shell> mkdir installation_directory
   Shell> cp path/to/installation_zip installation_directory
   Shell> cd installation_directory
   Shell> unzip installation_zip
   ```

   If you are on Linux or UNIX, run:

   ```
   Shell> tar -xf installation_tar
   ```

   This downloads the files into several of the subdirectories Directory Structure .

2. Stay on the installation directory and bring up GGSCI to create the remaining subdirectories in the installation location.

```
Shell> ggsci
GGSCI> CREATE SUBDIRS
```

3. Create a Manager parameter file:

```
GGSCI> EDIT PARAM MGR
```

4. Specify a port for the Manager to listen on by using the editor to add a line to the Manager parameter file. For example:

```
PORT 7801
```

5. If you are on Windows and running Manager as a service, set the system variable `PATH` to include `jvm.dll`, then delete the Manager service and re-add it.
6. Go to GGSCI, start the Manager, and check to see that it is running:

```
GGSCI>START MGR
GGSCI>INFO MGR
```

> **Note:**
>
> To check for environmental variable problems locating the JVM at runtime:
>
> - Add the parameter `GETENV(PATH)` for Windows or `GETENV(LD_LIBRARY_PATH)` for UNIX to the Replicat parameter file.
> - Start the Replicat process
> - Check the output for the report using the GGSCI command: `SEND REPLICAT group_name REPORT`

Setting Up Oracle GoldenGate for Big Data

The various tasks that you need to preform to set up Oracle GoldenGate for Big Data integrations with Big Data targets.

Topics:

Java Environment Setup

The Oracle GoldenGate for Big Data integrations create an instance of the Java virtual machine at runtime. Oracle GoldenGate for Big Data requires that you install Oracle Java 8 JRE at a minimum.

Oracle recommends that you set the `JAVA_HOME` environment variable to point to Java 8 installation directory. Additionally, the Java Delivery process needs to load the `libjvm.so` and `libjsig.so` Java shared libraries. These libraries are installed as part of the JRE. The location of these shared libraries need to be resolved and the

appropriate environmental variable set to resolve the dynamic libraries needs to be set so the libraries can be loaded at runtime (that is, `LD_LIBRARY_PATH`, `PATH`, or `LIBPATH`).

Properties Files

There are two Oracle GoldenGate properties files required to run the Oracle GoldenGate Java Deliver user exit (alternatively called the Oracle GoldenGate Java Adapter). It is the Oracle GoldenGate Java Delivery that hosts Java integrations including the Big Data integrations. A Replicat properties file is required in order to run either process. The required naming convention for the Replicat file name is the *`process_name`*`.prm`. The exit syntax in the Replicat properties file provides the name and location of the Java Adapter properties file. It is the Java Adapter properties file that contains the configuration properties for the Java adapter include GoldenGate for Big Data integrations. The Replicat and Java Adapters properties files are required to run Oracle GoldenGate for Big Data integrations.

Alternatively the Java Adapters properties can be resolved using the default syntax, *`process_name`*`.properties`. It you use the default naming for the Java Adapter properties file then the name of the Java Adapter properties file can be omitted from the Replicat properties file.

Samples of the properties files for Oracle GoldenGate for Big Data integrations can be found in the subdirectories of the following directory:

`GoldenGate_install_dir``/AdapterExamples/big-data`

Grouping Transactions

The principal way to improve performance in Oracle GoldenGate for Big Data integrations is using transaction grouping. In transaction grouping, the operations of multiple transactions are grouped together in a single larger transaction. The application of a larger grouped transaction is typically much more efficient than the application of individual smaller transactions. Transaction grouping is possible with the Replicat process discussed in Running with Replicat.

Configuring GoldenGate for Big Data

This section describes how to configure GoldenGate for Big Data Handlers.

Topics:

Running with Replicat

This section explains how to run the Java Adapter with the Oracle GoldenGate Replicat process. It includes the following sections:

Topics:

Configuring Replicat

The following is an example of how you can configure a Replicat process properties file for use with the Java Adapter:

```
REPLICAT hdfs
TARGETDB LIBFILE libggjava.so SET property=dirprm/hdfs.properties
--SOURCEDEFS ./dirdef/dbo.def
DDL INCLUDE ALL
GROUPTRANSOPS 1000
MAPEXCLUDE dbo.excludetable
MAP dbo.*, TARGET dbo.*;
```

The following is explanation of these Replicat configuration entries:

`REPLICAT hdfs` - The name of the Replicat process.

`TARGETDB LIBFILE libggjava.so SET property=dirprm/hdfs.properties` - Sets the target database as you exit to `libggjava.so` and sets the Java Adapters property file to `dirprm/hdfs.properties`.

`--SOURCEDEFS ./dirdef/dbo.def` - Sets a source database definitions file. It is commented out because Oracle GoldenGate trail files provide metadata in trail.

`GROUPTRANSOPS 1000` - Groups 1000 transactions from the source trail files into a single target transaction. This is the default and improves the performance of Big Data integrations.

`MAPEXCLUDE dbo.excludetable` - Sets the tables to exclude.

`MAP dbo.*, TARGET dbo.*;` - Sets the mapping of input to output tables.

Adding the Replicat Process

The command to add and start the Replicat process in `ggsci` is the following:

```
ADD REPLICAT hdfs, EXTTRAIL ./dirdat/gg
START hdfs
```

Replicat Grouping

The Replicat process provides the Replicat configuration property, `GROUPTRANSOPS`, to control transaction grouping. By default, the Replicat process implements transaction grouping of 1000 source transactions into a single target transaction. If you want to turn off transaction grouping then the `GROUPTRANSOPS` Replicat property should be set to `1`.

Replicat Checkpointing

In addition to the Replicat checkpoint file ,`.cpr`, an additional checkpoint file, `dirchk/`*`group`*`.cpj`, is created that contains information similar to `CHECKPOINTTABLE` in Replicat for the database.

Initial Load Support

Replicat can already read trail files that come from both the online capture and initial load processes that write to a set of trail files. In addition, Replicat can also be configured to support the delivery of the special run initial load process using `RMTTASK` specification in the Extract parameter file. For more details about configuring the direct load, see Loading Data with an Oracle GoldenGate Direct Load.

Unsupported Replicat Features

The following Replicat features are not supported in this release:

- `BATCHSQL`
- `SQLEXEC`
- Stored procedure
- Conflict resolution and detection (CDR)

Mapping Functionality

The Oracle GoldenGate Replicat process supports mapping functionality to custom target schemas. You must use the Metadata Provider functionality to define a target schema or schemas, and then use the standard Replicat mapping syntax in the Replicat configuration file to define the mapping. For more information about the Replicat mapping syntax in the Replication configuration file, see Mapping and Manipulating Data.

Logging

Logging is essential to troubleshooting Oracle GoldenGate for Big Data integrations with Big Data targets. This section covers how Oracle GoldenGate for Big Data integration log and the best practices for logging.

Topics:

Replicat Process Logging

Oracle GoldenGate for Big Data integrations leverage the Java Delivery functionality described in the Delivering Java Messages. In this setup, either a Oracle GoldenGate Replicat process loads a user exit shared library. This shared library then loads a Java virtual machine to thereby interface with targets providing a Java interface. So the flow of data is as follows:

Replicat Process —>User Exit—> Java Layer

It is important that all layers log correctly so that users can review the logs to troubleshoot new installations and integrations. Additionally, if you have a problem that requires contacting Oracle Support, the log files are a key piece of information to be provided to Oracle Support so that the problem can be efficiently resolved.

A running Replicat process creates or appends log files into the *GoldenGate_Home*`/dirrpt` directory that adheres to the following naming convention: *process_name*`.rpt`. If a problem is encountered when deploying a new Oracle GoldenGate process, this is likely the first log file to examine for problems. The Java layer is critical for integrations with Big Data applications.

Java Layer Logging

The Oracle GoldenGate for Big Data product provides flexibility for logging from the Java layer. The recommended best practice is to use Log4j logging to log from the Java layer. Enabling simple Log4j logging requires the setting of two configuration values in the Java Adapters configuration file.

```
gg.log=log4j
gg.log.level=INFO
```

These `gg.log` settings will result in a Log4j file to be created in the *GoldenGate_Home*`/dirrpt` directory that adheres to this naming convention, *process_name_log level*`_log4j.log`. The supported Log4j log levels are in the following list in order of increasing logging granularity.

- `OFF`
- `FATAL`
- `ERROR`
- `WARN`
- `INFO`
- `DEBUG`
- `TRACE`

Selection of a logging level will include all of the coarser logging levels as well (that is, selection of `WARN` means that log messages of `FATAL`, `ERROR` and `WARN` will be written to the log file). The Log4j logging can additionally be controlled by separate Log4j properties files. These separate Log4j properties files can be enabled by editing the `bootoptions` property in the Java Adapter Properties file. These three example Log4j properties files are included with the installation and are included in the classpath:

```
log4j-default.properties
log4j-debug.properites
log4j-trace.properties
```

You can modify the `bootoptions`in any of the files as follows:

```
javawriter.bootoptions=-Xmx512m -Xms64m -Djava.class.path=.:ggjava/ggjava.jar -
Dlog4j.configuration=samplelog4j.properties
```

You can use your own customized Log4j properties file to control logging. The customized Log4j properties file must be available in the Java classpath so that it can be located and loaded by the JVM. The contents of a sample custom Log4j properties file is the following:

```
# Root logger option
log4j.rootLogger=INFO, file

# Direct log messages to a log file
log4j.appender.file=org.apache.log4j.RollingFileAppender

log4j.appender.file.File=sample.log
log4j.appender.file.MaxFileSize=1GB
log4j.appender.file.MaxBackupIndex=10
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L -
%m%n
```

There are two important requirements when you use a custom Log4j properties file. First, the path to the custom Log4j properties file must be included in the `javawriter.bootoptions` property. Logging initializes immediately when the JVM is initialized while the contents of the `gg.classpath` property is actually appended to the `classloader` after the logging is initialized. Second, the `classpath` to correctly load a properties file must be the directory containing the properties file without wildcards appended.

Schema Evolution and Metadata Change Events

The Metadata in trail is a feature that allows seamless runtime handling of metadata change events by Oracle GoldenGate for Big Data, including schema evolution and schema propagation to Big Data target applications. The `NO_OBJECTDEFS` is a sub-parameter of the Extract and Replicat `EXTTRAIL` and `RMTTRAIL` parameters that lets you suppress the important metadata in trail feature and revert to using a static metadata definition.

The Oracle GoldenGate for Big Data Handlers and Formatters provide functionality to take action when a metadata change event is encountered. The ability to take action in the case of metadata change events depends on the metadata change events being available in the source trail file. Oracle GoldenGate supports metadata in trail and the propagation of DDL data from a source Oracle Database. If the source trail file does not have metadata in trail and DDL data (metadata change events) then it is not possible for Oracle GoldenGate for Big Data to provide and metadata change event handling.

Configuration Property `CDATA[]` Wrapping

The GoldenGate for Big Data Handlers and Formatters support the configuration of many parameters in the Java properties file, the value of which may be interpreted as

white space. The configuration handling of the Java Adapter trims white space from configuration values from the Java configuration file. This behavior of trimming whitespace may be desirable for some configuration values and undesirable for other configuration values. Alternatively, you can wrap white space values inside of special syntax to preserve the whites pace for selected configuration variables. GoldenGate for Big Data borrows the XML syntax of `CDATA[]` to preserve white space. Values that would be considered to be white space can be wrapped inside of `CDATA[]`.

The following is an example attempting to set a new-line delimiter for the Delimited Text Formatter:

```
gg.handler.{name}.format.lineDelimiter=\n
```

This configuration will not be successful. The new-line character is interpreted as white space and will be trimmed from the configuration value. Therefore the `gg.handler` setting effectively results in the line delimiter being set to an empty string.

In order to preserve the configuration of the new-line character simply wrap the character in the `CDATA[]` wrapper as follows:

```
gg.handler.{name}.format.lineDelimiter=CDATA[\n]
```

Configuring the property with the `CDATA[]` wrapping preserves the white space and the line delimiter will then be a new-line character.

Using Regular Expression Search and Replace

You can perform more powerful search and replace operations of both schema data (catalog names, schema names, table names, and column names) and column value data, which are separately configured. Regular expressions (regex) are characters that customize a search string through pattern matching. You can match a string against a pattern or extract parts of the match. Oracle GoldenGate for Big Data uses the standard Oracle Java regular expressions package, `java.util.regex`. For more information, see "Regular Expressions" in the Base Definitions volume of **The Single UNIX Specification, Version 4**.

Topics:

- Using Schema Data Replace
- Using Content Data Replace

Using Schema Data Replace

You can replace schema data using the `gg.schemareplaceregex` and `gg.schemareplacestring` properties. Use `gg.schemareplaceregex` to set a regular expression, and then use it to search catalog names, schema names, table names, and column names for corresponding matches. Matches are then replaced with the content of the `gg.schemareplacestring` value. The default value of `gg.schemareplacestring` is an empty string or `""`.

For example, some system table names start with a dollar sign like `$mytable`. You may want to replicate these tables even though most Big Data targets do not allow dollar signs in table names. To remove the dollar sign, you could configure the following replace strings:

```
gg.schemareplaceregex=[$]
gg.schemareplacestring=
```

The resulting example of searched and replaced table name is `mytable`. These properties also support `CDATA[]` wrapping to preserve whitespace in the value of configuration values. So the equivalent of the preceding example using `CDATA[]` wrapping use is:

```
gg.schemareplaceregex=CDATA[[$]]
gg.schemareplacestring=CDATA[]
```

The schema search and replace functionality supports using multiple search regular expressions and replacements strings using the following configuration syntax:

```
gg.schemareplaceregex=some_regex
gg.schemareplacestring=some_value
gg.schemareplaceregex1=some_regex
gg.schemareplacestring1=some_value
gg.schemareplaceregex2=some_regex
gg.schemareplacestring2=some_value
```

Using Content Data Replace

You can replace content data using the `gg.contentreplaceregex` and `gg.contentreplacestring` properties to search the column values using the configured regular expression and replace matches with the replacement string. For example, this is useful to replace line feed characters in column values. If the delimited text formatter is used then line feeds occurring in the data will be incorrectly interpreted as line delimiters by analytic tools.

You can configure *n* number of content replacement regex search values. The regex search and replacements are done in the order of configuration. Configured values must follow a given order as follows:

```
gg.contentreplaceregex=some_regex
gg.contentreplacestring=some_value
gg.contentreplaceregex1=some_regex
gg.contentreplacestring1=some_value
gg.contentreplaceregex2=some_regex
gg.contentreplacestring2=some_value
```

Configuring a subscript of 3 without a subscript of 2 would cause the subscript 3 configuration to be ignored.

> **NOT_SUPPORTED:**
>
> Regular express searches and replacements require computer processing and can reduce the performance of the Oracle GoldenGate for Big Data process.

To replace line feeds with a blank character you could use the following property configurations:

```
gg.contentreplaceregex=[\n]
gg.contentreplacestring=CDATA[ ]
```

This changes the column value from:

```
this is
me
```

to :

```
this is me
```

Both values support `CDATA` wrapping. The second value must be wrapped in a `CDATA[]` wrapper because a single blank space will be interpreted as whitespace and trimmed by the Oracle GoldenGate for Big Data configuration layer. In addition, you can configure multiple search a replace strings. For example, you may also want to trim leading and trailing white space out of column values in addition to trimming line feeds from:

```
^\\s+|\\s+$

gg.contentreplaceregex1=^\\s+|\\s+$
gg.contentreplacestring1=CDATA[]
```

Scaling Oracle GoldenGate for Big Data Delivery

Oracle GoldenGate for Big Data supports breaking down the source trail files into either multiple Replicat processes or by using Coordinated Delivery to instantiate multiple Java Adapter instances inside a single Replicat process to improve throughput.. This allows you to scale Oracle GoldenGate for Big Data delivery.

There are some cases where the throughput to Oracle GoldenGate for Big Data integration targets is not sufficient to meet your service level agreements even after you have tuned your Handler for maximum performance. When this occurs, you can configure parallel processing and delivery to your targets using one of the following methods:

- Multiple Replicat processes can be configured to read data from the same source trail files. Each of these Replicat processes are configured to process a subset of the data in the source trail files so that all of the processes collectively process the source trail files in their entirety. There is no coordination between the separate Replicat processes using this solution.
- Oracle GoldenGate Coordinated Delivery can be used to parallelize processing the data from the source trail files within a single Replicat process. This solution involves breaking the trail files down into logical subsets for which each configured subset is processed by a different delivery thread. For more information about Coordinated Delivery, see https://blogs.oracle.com/dataintegration/entry/goldengate_12c_coordinated_replicat.

With either method, you can split the data into parallel processing for improved throughput. Oracle recommends breaking the data down in one of the following two ways:

- Splitting Source Data By Source Table –Data is divided into subsections by source table. For example, Replicat process 1 might handle source tables table1 and table2, while Replicat process 2 might handle data for source tables table3 and table2. Data is split for source table and the individual table data is not subdivided.
- Splitting Source Table Data into Sub Streams – Data from source tables is split. For example, Replicat process 1 might handle half of the range of data from source table1, while Replicat process 2 might handler the other half of the data from source table1.

Additional limitations:

- Parallel apply is *not* supported.
- The `BATCHSQL` parameter not supported.

Example 1-3 Scaling Support for the Oracle GoldenGate for Big Data Handlers

Handler Name	Splitting Source Data By Source Table	Splitting Source Table Data into Sub Streams
Cassandra	Supported	Supported when: • Required target tables in Cassandra are pre-created. • Metadata change events do not occur.
Elastic Search		
Flume	Supported	Supported for formats that support schema propagation, such as Avro. This is less desirable due to multiple instances feeding the same schema information to the target.
HBase	Supported when all required HBase namespaces are pre-created in HBase.	Supported when: • All required HBase namespaces are pre-created in HBase. • All required HBase target tables are pre-created in HBase. Schema evolution is not an issue because HBase tables have no schema definitions so a source metadata change does not require any schema change in HBase. • The source data does not contain any truncate operations.

Handler Name	Splitting Source Data By Source Table	Splitting Source Table Data into Sub Streams
HDFS	Supported	Supported with some restrictions. • You must select a naming convention for generated HDFS files wherethe file names do not collide. Colliding HDFS file names results in a Replicat abend. When using coordinated apply it is suggested that you configure `${groupName}` as part of the configuration for the `gg.handler.name.fileNameMappingTemplate` property . The `${groupName}` template resolves to the Replicat name concatenated with the Replicat thread number, which provides unique naming per Replicat thread. • Schema propagatation to HDFS and Hive integration is *not* currently supported.
JDBC	Supported	Supported
Kafka	Supported	Supported for formats that support schema propagation, such as Avro. This is less desirable due to multiple instances feeding the same schema information to the target.
Kafka Connect	Supported	Supported
Kinesis Streams	Supported	Supported
MongoDB	Supported	Supported

Using Identities in Oracle GoldenGate Credential Store

The Oracle GoldenGate credential store manages user IDs and their encrypted passwords (together known as credentials) that are used by Oracle GoldenGate processes to interact with the local database. The credential store eliminates the need to specify user names and clear-text passwords in the Oracle GoldenGate parameter files. An optional alias can be used in the parameter file instead of the user ID to map to a userid and password pair in the credential store. The credential store is implemented as an auto login wallet within the Oracle Credential Store Framework (CSF). The use of an LDAP directory is not supported for the Oracle GoldenGate credential store. The auto login wallet supports automated restarts of Oracle

GoldenGate processes without requiring human intervention to supply the necessary passwords.

In Oracle GoldenGate for Big Data, you specify the alias and domain in the property file not the actual user ID or password. User credentials are maintained in secure wallet storage.

Topics:

Creating a Credential Store

You can create a credential store for your Big Data environment.

Run the GGSCI `ADD CREDENTIALSTORE` command to create a file called `cwallet.sso` in the `dircrd/` subdirectory of your Oracle GoldenGate installation directory (the default).

You can the location of the credential store (`cwallet.sso` file by specifying the desired location with the `CREDENTIALSTORELOCATION` parameter in the `GLOBALS` file.

For more information about credential store commands, see *Reference for Oracle GoldenGate*.

> **Note:**
>
> Only one credential store can be used for each Oracle GoldenGate instance.

Adding Users to a Credential Store

After you create a credential store for your Big Data environment, you can added users to the store.

Run the GGSCI `ALTER CREDENTIALSTORE ADD USER` *`userid`* `PASSWORD` *`password`* `[ALIAS` *`alias`*`] [DOMAIN` *`domain`*`]` command to create each user, where:

- *`userid`* is the user name. Only one instance of a user name can exist in the credential store unless the `ALIAS` or `DOMAIN` option is used.
- *`password`* is the user's password. The password is echoed (not obfuscated) when this option is used. If this option is omitted, the command prompts for the password, which is obfuscated as it is typed (recommended because it is more secure).
- *`alias`* is an alias for the user name. The alias substitutes for the credential in parameters and commands where a login credential is required. If the `ALIAS` option is omitted, the alias defaults to the user name.

For example:

```
ALTER CREDENTIALSTORE ADD USER scott PASSWORD tiger ALIAS scsm2 domain ggadapters
```

For more information about credential store commands, see *Reference for Oracle GoldenGate*.

Configuring Properties to Access the Credential Store

The Oracle GoldenGate Java Adapter properties file requires specific syntax to resolve user name and password entries in the Credential Store at runtime. For resolving a user name the syntax is the following:

```
ORACLEWALLETUSERNAME alias domain_name
```

For resolving a password the syntax required is the following:

```
ORACLEWALLETPASSWORD alias domain_name
```

The following example illustrate how to configure a Credential Store entry with an alias of `myalias` and a domain of `mydomain`.

> **Note:**
>
> With HDFS Hive JDBC the user name and password is encrypted.

Oracle Wallet integration only works for configuration properties which contain the string username or password. For example:

```
gg.handler.hdfs.hiveJdbcUsername=ORACLEWALLETUSERNAME[myalias mydomain]
gg.handler.hdfs.hiveJdbcPassword=ORACLEWALLETPASSWORD[myalias mydomain]
```

Consider the user name and password entries as accessible values in the Credential Store. Any configuration property resolved in the Java Adapter layer (not accessed in the C user exit layer) can be resolved from the Credential Store. This allows you more flexibility to be creative in how you protect sensitive configuration entries.

2
Upgrading Oracle GoldenGate for Big Data

This chapter describes how to upgrade to Oracle GoldenGate for Big Data 12*c* (12.3.1.2) by downloading the product as described in Downloading Oracle GoldenGate for Big Data, and then choosing the upgrade paths that suits your environment.

After upgrading, you must then convert your Extract processes to Replicat processes.

> **Note:**
>
> There is no supported upgrade path from release 12.1.2 to 12.2.0.*x*.
>
> There is no supported upgrade path from release 12.2.0.1.0 to 12.2.0.1.1.

> **Note:**
>
> You must upgrade the Cassandra driver to version 3.3.1 if you are using a different version.

Topics:

Upgrading by Overwriting the Existing Installation

The most straightforward upgrade path is to copy the Oracle GoldenGate for Big Data 12*c* (12.3.1.*x*) files into the existing 12*c* (12.3.1.*x* or 12.2.0.*x*) installation directory. Overwriting the product files is possible because there is neither structural nor package name changes in 12.3.1.*x*. The 12.3.1.*x* and 12.2.0.*x* handler and formatter configurations are also compatible with the 12.3.1.*x* release.

1. (Source and target systems) Back up the current Oracle GoldenGate for Big Data installation directory on the source and target systems, and any working directories that you have installed on a shared drive in a cluster (if applicable).
2. (Source and target systems, as applicable) Expand version 12*c* (12.3.1.*x* or 12.2.0.*x*) of Oracle GoldenGate into a new directory on each system (not the current Oracle GoldenGate directory). Do not create the sub-directories; just complete the steps to the point where the installation files are expanded.
3. (Source system) Stop user activity on objects in the Oracle GoldenGate configuration.

4. Create an installation directory with *no* spaces in its name.
5. Extract the ZIP file into this new installation directory, which divides the files into several subdirectories.
6. Copy the Oracle GoldenGate for Big Data 12*c* (12.3.1.*x*) files into the existing 12*c* (12.3.1.*x* or 12.2.0.*x*) installation directory.
7. Start the Replicat processes and verify that they are running.

```
GGSCI> START MANAGER
GGSCI> START REPLICAT group_name
GGSCI> INFO REPLICAT group_name
GGSCI> VIEW REPORT group_name
```

Certain 12.2.0.*x* JAR files are still present after the overwriting process though the new 12.3.1.*x* JAR files are used.

Upgrading by Installing into a New Directory

Use the following steps to install the Oracle GoldenGate for Big Data 12*c* (12.3.1.*x*) files into a new installation directory.

1. (Source and target systems) Back up the current Oracle GoldenGate for Big Data installation directory on the source and target systems, and any working directories that you have installed on a shared drive in a cluster (if applicable).
2. (Source and target systems, as applicable) Expand version 12*c* (12.3.1.*x* or 12.2.0.1) of Oracle GoldenGate into a new directory on each system (not the current Oracle GoldenGate directory). Do not create the sub-directories; just complete the steps to the point where the installation files are expanded.
3. (Source system) Stop user activity on objects in the Oracle GoldenGate configuration.
4. Create an installation directory with *no* spaces in its name.
5. Extract the ZIP file into this new installation directory, which divides the files into several subdirectories.
6. In the installation directory, start GGSCI to create the remaining subdirectories in the installation location.

```
$ ggsci
GGSCI> CREATE SUBDIRS
```

7. Copy all of the `dirprm` files from your existing installation into the `dirprm` directory in the new installation location.

> **Note:**
>
> All of your configuration files must be in the `dirprm` directory. If you have property files, Velocity templates, or other configuration files in a location other than `dirprm` in your old installation, then you must copy them to the `dirprm` directory in the new installation.

8. Copy all of the `dirdef` files from your existing installation into the `dirdef` directory in the new installation location.

9. If you have data files stored in the 12.3.1.*x* or 12.2.0.*x* installation `dirdat` directory, then copy or move the existing trail files to the `dirdat` directory of the new installation.
10. If you have additional JAR files or other custom files in your 12.3.1.*x* or 12.2.0.*x* installation, then copy them to the new installation directory.
11. Configure the Replicat processes in the new installation directory by starting GGSCI and adding the Replicat and naming the trails.

    ```
    GGSCI> ADD REPLICAT group_name, EXTTRAIL trail_name, ...
    GGSCI> ALTER group_name EXTSEQNO seqno EXTRBA rba
    ```

    Optionally, you could alter the starting position of Replicat processing as needed.
12. Start the Replicat processes and verify that they are running.

    ```
    GGSCI> START MANAGER
    GGSCI> START REPLICAT group_name
    GGSCI> INFO REPLICAT group_name
    GGSCI> VIEW REPORT group_name
    ```
13. Modify the source system to write to the new Oracle GoldenGate for Big Data installation directory:
    a. (Optional) Upgrade the source database Oracle GoldenGate capture following the upgrade procedure for your database platform.
    b. Configure the source database capture to write to the new Oracle GoldenGate for Big Data 12.3.1.*x* installation `dirdat` directory.
    c. When the old Oracle GoldenGate for Big Data installation has processed all its data, switch over to the process that will send data to the new location.

Switching Existing Extract Processes to Replicat Processes

In previous releases, you could use an Extract and pump process to write to your Big Data targets. In this release, this solution is deprecated so you must use Replicat.

A typical Extract configuration is similar to:

```
EXTRACT mygroup
SOURCEDEFS path/to/source/def/file
CUSEREXIT libggjava_ue.so CUSEREXIT PASSTHRU INCLUDEUPDATEBEFORES
GETUPDATEBEFORES
TABLE *.*;
```

With Replicat the preceding configuration would be:

```
REPLICAT mygroup
SOURCEDEFS path/to/source/def/file
TARGETDB LIBFILE libggjava.so SET property=/path/to/properties/file
MAP *.* TARGET *.*;
```

The same properties file works in the Replicat configuration so you do not need to change the properties file.

To complete this process, you must add the Replicat group, start the process, and verify that it is running:

```
GGSCI> ADD REPLICAT group_name, EXTTRAIL trail_name, ...
GGSCI> ALTER group_name EXTSEQNO seqno EXTRBA rbaGGSCI> START REPLICAT group_name
```

```
GGSCI> INFO REPLICAT group_name
GGSCI> VIEW REPORT group_name
```