Oracle® GoldenGate

Administering Oracle GoldenGate Veridata

12*c* (12.2.1.2.0)
F10936-01
December 2018

Oracle GoldenGate Administering Oracle GoldenGate Veridata, 12*c* (12.2.1.2.0)

F10936-01

Contents

Index

Preface

This document describes how to configure and administer Oracle GoldenGate Veridata.

Audience

This document is intended for installers and system administrators who are installing, configuring and running Oracle GoldenGate Veridata.

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.

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

Conventions

The following text conventions are used in this document:

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

1

Introduction to Oracle GoldenGate Veridata

Oracle GoldenGate Veridata provides an overview of roles and interactions of the components, how to configure components. It compares one set of data to another and identifies data that is out-of-sync, and allows you to repair that data.

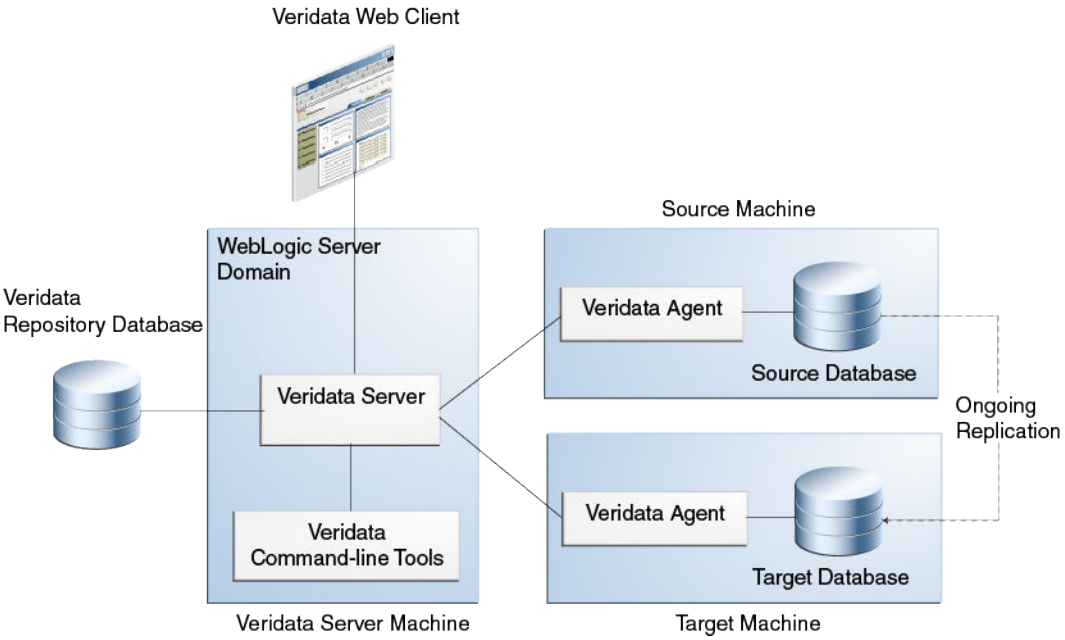
This chapter includes the following sections:

Oracle Oracle GoldenGate Veridata Architecture

Oracle Oracle GoldenGate Veridata compares one set of data to another, identifies data that is out-of-sync, and enables you to repair any out-of-sync data. Oracle GoldenGate Veridata supports high-volume, 24x7 heterogeneous replication environments where downtime to compare data sets is not an option. By accounting for data that is being replicated while a comparison takes place, Oracle Oracle GoldenGate Veridata can run concurrently with data transactions and replication and still produce an accurate comparison report.

Oracle Oracle GoldenGate Veridata maps column data types across different types of databases automatically. For more information about this feature in Oracle GoldenGate Veridata web user interface, see the online help. Alternatively, you can map columns manually if automatic mapping does not accommodate format differences in a heterogeneous environment. To manually upload an XML file, use the `veridata_import` utility. For more information, see Using the Veridata Import and Export Utilities.

Figure 1-1 Oracle GoldenGate Veridata Architecture



Oracle GoldenGate Veridata Server

The Oracle GoldenGate Veridata Server performs the following functions:

- Coordinate the execution of Oracle GoldenGate Veridata tasks
- Sort rows (optional)
- Compare data
- Confirm out-of-sync data
- Produce a report for review

Oracle GoldenGate Veridata Web User Interface

The Oracle GoldenGate Veridata web user interface (UI) is a browser-based graphical user interface for these activities:

- Configure comparison objects and rules
- Initiate comparisons
- Review the status and output of comparisons
- Repair out-of-sync data
- Review out-of-sync data

Oracle GoldenGate Veridata Repository

The Oracle GoldenGate Veridata repository is a collection of database objects that persists configuration information to disk, saving it permanently as a user environment.

> **Note:**
>
> Out-of-sync data is not stored in the repository. This data is stored on the server's file system.

Oracle GoldenGate Veridata Agent

The Oracle GoldenGate Veridata Agent (agent) executes the following database-related requests on behalf of the Oracle GoldenGate Veridata server:

- Hash rows for initial comparison
- Fetch and update rows to repair out-of-sync data
- Return column-level details for out-of-sync rows

Oracle GoldenGate Veridata Manager

The Oracle GoldenGate Veridata Manager is part of the C-code agent that is required for the NonStop platform. It controls the agent process.

The Manager is not used in a Java agent, which is used for the other databases that are supported by Oracle GoldenGate Veridata.

Oracle GoldenGate Veridata Command Line Utilities

Oracle GoldenGate Veridata includes the following command-line utilities:

Table 1-1 Command-Line Utilities

Name	Description
`vericom`	Enables you to run comparisons by using automated programs. See **Managing Identities and Comparing Data**.
`veridata_import`	Maps comparison objects and rules in an XML file and imports it into the repository. See Using the Veridata Import and Export Utilities .
`veridata_export`	Maps comparison objects and rules in the repository and exports them to an XML file. See Using the Veridata Import and Export Utilities .
`veridata_param_process`	Helps to use the Oracle GoldenGate parameter files. See Introduction to Oracle GoldenGate Veridata.
`reportutil`	Supports viewing encrypted report files and out-of-sync data.

Configuring Single Sign-on for Oracle GoldenGate Veridata

Oracle GoldenGate Veridata 12*c* (12.2.1) supports single sign-on (SSO) for authentication. When you configure SSO, set the SSO properties of the server and configure the sign-out URL for the SSO session.

To configure SSO for Oracle GoldenGate Veridata server:

1. Set the `web.singleSignOutUrl` parameter in the `DOMAIN_HOME/config/veridata/ veridata.cfg` file.

2. Run the `configureVeridata` script: `DOMAIN_HOME/veridata/bin/configureVeridata.sh -pUweb.singleSignOutUrl=Single sign out URL`.

If your domain no longer uses SSO, then remove the SSO logout configuration: `DOMAIN_HOME/veridata/bin/configureVeridata.sh -pUweb.singleSignOutUrl=default`.

The parameter usage is explained in the `veridata.cfg` file:

```
# (web.singleSignOutUrl) as
# web.singleSignOutUrl - Specify the Single Sign Out URL here:
# Formats: /oamsso/logout.html?end_url=/veridata
# http://myoamserverhost:port/oam/server/logout?end_url=http://
my.veridata.site.com:veridata-port/veridata
# http://myoamserverhost:port/oamsso/logout.html?end_url==http://
my.veridata.site.com:veridata-port/veridata

# This URL must conform to the grammar in RFC 2396, except the few deviations
mentioned in the java documentation for construction of a URI by parsing the given
string.

web.singleSignOutUrl default
```

Comparing Data with Oracle GoldenGate Veridata

This section explains how to configure the objects that are to be compared and how Oracle GoldenGate Veridata processes comparisons.

Oracle GoldenGate Veridata Comparison Objects

To begin using Oracle GoldenGate Veridata, you create the following objects that help you manage your work and identify the data that you want to compare:

- Data source connections
- Groups
- Compare Pairs
- Profiles
- Jobs

To order rows for comparison, Oracle GoldenGate Veridata relies on a unique identifier (primary key, unique key, user-defined key).

For more information, see the *Oracle GoldenGate Veridata Online Help*.

Satisfying Uniqueness Requirements

Oracle Oracle GoldenGate Veridata relies on a unique identifier to order rows for comparison.

- **Primary Key:** By default, Oracle GoldenGate Veridata uses the primary key if one is available.
- **Unique Key:** If no primary key is defined, then Oracle GoldenGate Veridata uses the smallest unique index.

- **User-defined Key:** If a table or file has doesn't have a primary or a unique key, then you can define an existing index or a set of columns for comparison purposes when you define a compare pair. However, although primary or unique keys can be mapped automatically, user-defined keys must be mapped manually. If you prefer a different ordering method, you can also use a user-defined key to override existing keys or indexes.

For more information about choosing and mapping keys for comparison, see the online help.

How Oracle GoldenGate Veridata Compares Data

Comparison activities consist of an initial comparison step and a confirmation step. To change the steps' aspects, change the parameters in the Oracle GoldenGate Veridata web user interface.

Initial Comparison Step

In the *initial comparison* (or *row hash*) step, rows are retrieved from the source and target tables with a query. If the source and target databases are of different types, the columns are converted to a standardized data type format for accurate comparison. By default, Oracle GoldenGate Veridata compares rows by comparing all columns of the primary key literally (value-for-value) and by using a hash value for all non-key columns. The unique digital signature that is used to calculate the hash value shrinks the data to be transferred over the network for the comparison. The signature still provides a highly reliable (but not absolute) and efficient mechanism for determining whether two rows contain the same or different column values.

To ensure that you discover out-of-sync rows, you can configure Oracle GoldenGate Veridata to compare non-key rows column-for-column. Full-column comparisons reduce the processing performance in proportion to the number of columns, and they increase network usage.

For all supported databases, you can use the delta processing performance feature if you are using server-side sorting. For this processing, Oracle GoldenGate Veridata detects which data blocks were modified since a previous comparison and only compares the rows in those blocks. Rows in unchanged blocks are skipped. The default is to compare all rows.

Delta processing consists of two steps:

- Collect the base modification time of the previous run for subsequent delta comparisons. This step is always included when you enable delta processing for a compare pair.
- Compare data that was modified since the base comparison and use the information that was collected in the first step. To enable this step, click **Enable Delta Processing** on the Compare Pair Configuration page and the Run/Execute Job page in the Oracle GoldenGate Veridata web user interface. Disable delta comparison when modifications, such as table reorganizations, invalidate the collected delta base information.

For more information about delta processing, see "Using Delta Processing" in *Oracle GoldenGate VeridataUser's Guide* or online help.

After the initial comparison, rows that appear to be out-of-sync are stored in a maybe out-of-sync (MOOS) queue in memory, because the comparison is inconclusive. When

a replication is working concurrently with a comparison, especially if there is replication latency, rows can appear to be out-of-sync. However, the current data is in flight (somewhere in the replication flow), and replication resynchronizes them.

Confirmation Step

The *confirmation*, or *confirm-out-of-sync (COOS)*, step ensures accurate results by confirming the row status in a changing environment. This step involves predicated queries on the source or target database by using the rows extracted from the MOOS queue. The status is evaluated as one of the following:

- *in-flight*: The row was out-of-sync in the initial comparison step, but it was updated. It is assumed that replication or another mechanism applied the change, but Oracle GoldenGate Veridata was unable to confirm that the rows were in-sync.
- *in-sync*: The source row values were applied to the target row by replication or another method. An in-sync status does not guarantee that the rows are synchronized at any particular moment if the underlying tables are continuously changing, but it does indicate that replication is working.
- *persistently out-of-sync*: The row was not updated since the initial comparison step, and can be assumed to be out-of-sync.

By default, confirmation processing occurs in a thread that is parallel to the initial comparison step, but the confirmation of each row waits until a specified replication latency threshold expires. For example, if latency is 60 seconds, and the initial comparison step revealed an out-of-sync row at 9:30, then the confirmation step for that row is not performed until 9:31 to allow replication to apply any change that was in-flight. After latency is accounted for, rows can be confirmed as persistently out-of-sync and are stored in one or more out-of-sync reports.

Viewing Comparison Results

When a job is completed, you can view the out-of-sync report and the comparison reports by using the Oracle GoldenGate Veridata web user interface or by viewing the files themselves.

If report encryption is enabled for the server, then use the reportutil tool to view the report files. See Encrypting Report Files. The Oracle GoldenGate Veridata web user interface automatically decrypts the file before displaying them.

You can store an OOS report in binary format, in XML format, or both (or none).

- **OOS file**: When stored in binary form, the OOS report contains out-of-sync comparison results that you use for viewing row differences in the Oracle GoldenGate Veridata web user interface. You can also use the report to recompare out-of-sync rows later. To recompare rows, select run options to execute another confirmation step. The step compares the current state of just those rows and then reports which rows remain out-of-sync after replication or another restorative procedure was applied.
- **OOSXML file**: When the OOS report is stored as XML, it is written to an OOSXML file and is stored to conform to an internal XML schema. Among the many advantages of XML, the most significant is that it can be easily manipulated by many tools. In its XML form, the file contains all information, including metadata, that is needed to select rows for resynchronization by external programs.

Each finished job, group, and compare pair generates a comparison report with the following type of information:

- Comparison parameters used
- Number of rows compared and out-of-sync
- Timing of the comparison
- Performance statistics
- Source and target data values

By default, the OOS files are located in the following subdirectories of the server installation directory. You can change the default location by specifying another path for the `server.veridata_data` property in the `veridata.cfg` file:

- **OOS files**: *`VERIDATA_DOMAIN_HOME`*`/veridata/reports/oos`
- **OOSXML files**: *`VERIDATA_DOMAIN_HOME`*`/veridata/reports/oosxml`

These subdirectories are further organized by run ID, job name, group name, and compare pair. In the OOSXML directory, the `.oosxml` files are the control files. The files with sequential file extensions are the OOSXML chunks. The XML data is spread into multiple files (called "chunks") for performance purposes. You can encrypt the comparison reports. For more information, see Encrypting Report Files.

2
Configuring Security

Oracle GoldenGate Veridata provides a safe, secure environment for your business data by using Secure Sockets Layer (SSL) and plain socket communications. You can control your security by managing passwords and encrypting the report files.
This chapter includes the following topics:

Overview of Oracle GoldenGate Veridata Security

When using Oracle GoldenGate Veridata, you select, view, and store data values from the tables or files of your business applications. Ensure to protect access to the following components:

- Files, programs, and directories in the Oracle GoldenGate Veridata installation directories. installation directories
- Data files that contain the results of data comparisons
- Oracle GoldenGate Veridata web user interface, where data values can be viewed

Configuring an SSL Connection between Oracle GoldenGate Veridata Server and Agents

Oracle GoldenGate Veridata supports SSL and plain socket communication between Oracle GoldenGate Veridata Server (server) and multiple Oracle GoldenGate Veridata Agents (agents) that are connected over a network. This section describes how to configure SSL and secure communication between the server and the agents.

> **Note:**
>
> The Oracle GoldenGate Veridata Agent for NonStop platforms doesn't support SSL communication.

In an SSL scenario, the server is considered the SSL Client and the agents are considered the SSL Servers. The server and the agents authenticate each other's identity. The data exchanged between the server and agent is also encrypted.

Configuring One-Way and Two-Way SSL Connections

SSL can be configured one-way or two-way in Oracle GoldenGate Veridata.

In a one-way SSL connection, the SSL client (Oracle GoldenGate Veridata Server) must trust the SSL server (Oracle GoldenGate Veridata Agent). In a two-way SSL connection, mutual trust is required between the SSL server and client. To enable SSL, you can use either self-signed certificates or Certificate Authority (CA) signed certificates.

To establish one-way SSL using self-signed certificates:

1. Create self-signed certificates for all agents.
2. Upload all agent certificates to the `VeridataWebTrustStore` of the server. See Managing the Server Identity and Trust Keystores.

To establish two-way SSL by using CA signed certificates:

1. Create self-signed certificates for all agents.
2. Upload all agent certificates to the `VeridataWebTrustStore` of the server.
3. Create a self-signed certificate for the identity store of the server.
4. Upload the server identity certificate to all agent truststores.

For more information about creating and importing certificates, see Creating Keystores and Self-Signed Certificates.

To establish one-way SSL by using CA signed certificates:

1. Use certificates issued by the same CA for all agents.
2. Trust the root CA certificate in the server.

To establish two-way SSL by using CA signed certificates:

1. Use certificates issued by the same CA for all agents.
2. Trust the root CA certificate in the server.
3. Use the certificate issued by a CA for the identity store of the server.
4. Trust the root CA certificate used in the previous step in the agent truststore.

Process for Enabling SSL

By default, the server and the agents are not enabled for SSL. If you decide to use SSL, you must enable the server and agent properties.

You must also create the identity and trust keystores. Create self-signed certificates if you are not using a CA certificate.

To establish an SSL connection between the server and the agent:

1. Configure SSL for the server. See Configuring SSL Settings for the Oracle GoldenGate Veridata Server.

2. Restart the server.
3. Shut down the agent. Configure SSL for the agent. See Configuring SSL Settings for the Oracle GoldenGate Veridata Agent.
    a. Obtain the agent-side keystores. See Managing the Server Identity and Trust Keystores.
    b. Configure the agent-side keystores in the agent configuration properties file.
4. Run `configure_agent_ssl.sh` and supply the password to the keystores configured in the agent configuration file. See Modifying the Oracle GoldenGate Veridata Agent Wallet.
5. Start the agent.
6. If the trust is established properly between agent keystores and their server counterpart in the Oracle Platform Security Services Keystore Service, then SSL communication is established.

Configuring SSL Settings for the Oracle GoldenGate Veridata Agent

By default, SSL is disabled for the agent. To configure SSL, edit the following properties in the `agent.properties` file for your agent.

Table 2-1 SSL Parameters in agent.properties file

Parameter	Description	Default Value
server.useSsl	Enables or disables SSL communication between the agent and the server. Possible values are: true: Enables SSL communication false: Disables SSL communication	false
server.use2WaySsl	Specifies whether the SSL communication false value is one way or two way. Options are: true: Uses two-way SSL communication false: Uses one-way SSL communication	false
server.identitystore.type	Specifies the type of keystore used for the SSL configuration.	JKS
server.identitystore.path	Specifies the path for the server identity keystore.	./certs/ serverIdentity.jks(Self Signed)
server.truststore.type	Specifies the type of truststore used for SSL configuration.	JKS
server.truststore.path	Specifies the path for the server truststore.	./certs/ serverTrust.jks(Self Signed)
server.identitystore.keyfactory.alg.name	Algorithm name of the keyfactory used for the SSL server identity store.	SunX509
server.truststore.keyfactory.alg.nam	Algorithm name of the keyfactory used for SSL server trust store.	SunX509

Table 2-1 (Cont.) SSL Parameters in agent.properties file

Parameter	Description	Default Value
server.ssl.algorithm.name	SSL algorithm name. **Note:** The value of this parameter must be same for the agent and the server.	TLS

Configuring SSL Settings for the Oracle GoldenGate Veridata Server

To enable SSL communication for all server-agent connections, you must set the SSL parameters in the `veridata.cfg` file located in the *`DOMAIN_HOME`*`/config/veridata` directory of your installation. Table 2-2 describes the various parameters that you must set in the `veridata.cfg` file for SSL communication.

To establish SSL communication only for certain connections, edit the connection properties in the Oracle GoldenGate Veridata web user interface. For more information, see the online help.

> **Note:**
>
> In addition to these settings, configure the server identity keystore and trust keystore by using the OPSS Keystore Service in the Oracle WebLogic Server domain. For more details, Managing the Server Identity and Trust Keystores.

Table 2-2 SSL Settings in the veridata.cfg file

Parameter	Description	Default Value
server.useSsl	Enables or disables SSL communication between the agent and the server. Possible values are: true: Enables SSL communication false: Disables SSL communication	false
server.ssl.client.identitystore.keyfactory.alg.name	Algorithm name of the keyfactory used for the SSL server identity store.	SunX509
server.ssl.client.truststore.keyfactory.alg.name	Algorithm name of the keyfactory used for the SSL server trust store.	SunX509
server.ssl.algorithm.name	SSL algorithm name. **Note:** The value of this parameter must be the same for the agent and the server.	TLS

Creating Keystores and Self-Signed Certificates

For mutual authentication and to establish SSL communication, the server and the agents should mutually trust the add certificates in the respective truststores.

This section explains how to create keystores and self-signed certificates by using the `keytool` utility that is available as part of the Java Runtime Environment (JRE). For more details about `keytool`, refer to the Java documentation at `http://docs.oracle.com/javase/7/docs/technotes/tools/#security`.

Creating an Identity Keystore with a Self-Signed Certificate

To create a `keystore` containing a self-signed certificate:

1. Enter

   ```
   keytool -genkey -keystore certs -keyalg rsa -alias vdt_alias -storepass
   server_ks_pwd -keypass server_pwd
   ```

2. At the prompts, enter the requested details about the certificate.

Building Server and Agent Keystores

1. Build the agent keystore:

   ```
   keytool -genkey -alias agent.server.keys -keyalg RSA -keystore
   agent.server.keystore -storepass ks_password -keypass keypwd
   ```

2. Export the agent certificate to a file, run the following `keytool` command:

   ```
   keytool -export -alias agent.server.keys -keystore agent.server.keystore -
   storepass ks_password -file agent.server.cer
   ```

3. Build the server keystore, run the following `keytool` command:

   ```
   keytool -genkey -alias vdt.web.client.keys -keyalg RSA -keystore
   vdt.web.client.keystore -storepass ks_password -keypass keypwd
   ```

4. Export the server certificate to a file, run the following `keytool` command:

   ```
   keytool -export -alias vdt.web.client.keys -keystore vdt.web.client.keystore -
   storepass ks_password -file vdt.web.client.cer
   ```

Importing Certificates to the Server and Agent Truststores

1. Import the server certificate to the agent truststore, enter:

   ```
   keytool -import -v -keystore agent.server.truststore -storepass ks_password -
   file vdt.web.client.cer
   ```

2. Import the agent certificate the server truststore, run the following `keytool` command:

   ```
   keytool -import -v -keystore vdt.web.client.truststore -storepass ks_password -
   file agent.server.cer
   ```

Examples

Example 1 Create an Agent ID Keystore

`keytool -genkey -alias vdt.agent.id -keyalg RSA -keystore vdtAgentID.jks -storepass changeit -keypass changeit -validity 365`

```
C:\java\Java8\jdk1.8.0_40\bin>keytool -genkey -alias vdt.agent.id -keyalg RSA -
keystore vdtAgentID.jks -storepass changeit -keypass c
```

```
hangeit -validity 365
What is your first and last name?
  [Unknown]:  COMPANY A
What is the name of your organizational unit?
  [Unknown]:  NA
What is the name of your organization?
  [Unknown]:  COMPANY A
What is the name of your City or Locality?
  [Unknown]:  USA
What is the name of your State or Province?
  [Unknown]:  USA
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=COMPANY A, OU=NA, O=COMPANY A, L=USA, ST=USA, C=US correct?
  [no]:  yes
```

keytool -export -alias vdt.agent.id -keystore vdtAgentID.jks -storepass changeit -file vdtAgentID.cer

C:\java\Java8\jdk1.8.0_40\bin>keytool -export -alias vdt.agent.id -keystore vdtAgentID.jks -storepass changeit -file *vdtAgentID.cer*

The certificate is stored in the *vdtAgentID.cer* file.

Example 2 Create a Server ID Keystore

keytool -genkey -alias vdt.server.id -keyalg RSA -keystore vdtServerID.jks -storepass changeit -keypass changeit -validity 365

```
C:\java\Java8\jdk1.8.0_40\bin>keytool -genkey -alias vdt.server.id -keyalg RSA -
keystore vdtServerID.jks -storepass changeit -keypass changeit -validity 365

What is your first and last name?
  [Unknown]:  ORACLE GOLDENGATE VERIDATA SERVER
What is the name of your organizational unit?
  [Unknown]:  NA
What is the name of your organization?
  [Unknown]:  COMPANY A
What is the name of your City or Locality?
  [Unknown]:  USA
What is the name of your State or Province?
  [Unknown]:  USA
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=COMPANY A, OU=NA, O=COMPANY A, L=USA, ST=USA, C=US correct?
[no]:  yes
```

keytool -export -alias vdt.server.id -keystore vdtServerID.jks -storepass changeit -file vdtServerID.cer

```
C:\java\Java8\jdk1.8.0_40\bin>keytool -export -alias vdt.server.id -keystore
vdtServerID.jks -storepass changeit -file vdtServerID.cer
```

The certificate is stored in the *vdtAgentID.cer* file.

Example 3 Create a TrustStore for Agent and Server

keytool -import -v -keystore vdtAgentTrust.jks -storepass changeit -file vdtServerID.cer -alias vdt.server.id

```
C:\java\Java8\jdk1.8.0_40\bin>keytool -import -v -keystore vdtAgentTrust.jks -
storepass changeit -file vdtServerID.cer -alias vdt.ser
ver.id
Owner: CN=ORACLE GOLDENGATE VERIDATA SERVER, OU=NA, O=COMPANY A, L=USA, ST=USA, C=US
Issuer: CN=ORACLE GOLDENGATE VERIDATA SERVER, OU=NA, O=COMPANY A, L=USA, ST=USA,
C=US
Serial number: 2aded02f
Valid from: Thu May 14 12:18:09 IST 2015 until: Fri May 13 12:18:09 IST 2016
Certificate fingerprints:
         MD5:  4E:7D:89:F7:C8:E8:64:37:E5:0C:D3:03:8F:3E:94:0A
         SHA1: 1B:00:9D:44:BD:73:6E:71:9D:44:56:4A:29:4E:F5:D7:1C:49:57:F3
         SHA256: 25:CB:77:3F:BC:5F:88:4B:09:D2:2D:C1:F8:E6:BA:70:DB:2B:
55:53:48:7D:BA:F1:A3:01:18:AB:AA:D1:56:6A
         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: EF C3 25 BB 83 4E 2D 0D   15 3D EF 50 F7 F2 D0 A6  ..%..N-..=.P....
0010: 94 5F 87 F2                                        ._..
]
]

Trust this certificate? [no]:  yes
Certificate was added to keystore
[Storing vdtAgentTrust.jks]
```

keytool -import -v -keystore vdtServerTrust.jks -storepass changeit -file vdtAgentID.cer -alias vdt.agent.id

```
C:\java\Java8\jdk1.8.0_40\bin>keytool -import -v -keystore vdtServerTrust.jks -
storepass changeit -file vdtAgentID.cer -alias vdt.age
nt.id
Owner: CN=COMPANY A, OU=NA, O=COMPANY A, L=USA, ST=USA, C=US
Issuer: CN=COMPANY A, OU=NA, O=COMPANY A, L=USA, ST=USA, C=US
Serial number: 6b590df2
Valid from: Thu May 14 12:08:00 IST 2015 until: Fri May 13 12:08:00 IST 2016
Certificate fingerprints:
         MD5:  3E:75:A3:96:40:60:10:96:DD:10:7B:4D:E4:3F:4C:04
         SHA1: D1:CC:EB:67:A1:C6:CD:CA:62:27:EA:F8:82:BF:AB:E4:E7:2B:45:6D
         SHA256: E7:20:CF:D4:48:E2:AE:1E:1C:C7:06:1A:B3:0A:17:1F:8F:
02:88:B7:A6:A0:5D:F7:12:BC:26:68:5B:C3:C9:C8
         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: C0 D5 02 D9 24 6F 58 F6   63 D7 34 D3 9D C4 9E 33  ....$oX.c.4....3
0010: FC 16 4E 5F                                        ..N_
]
]

Trust this certificate? [no]:  yes
```

```
Certificate was added to keystore
[Storing vdtServerTrust.jks]
```

Managing the Server Identity and Trust Keystores

Use the Oracle Platform Security Services keystore service as a repository for storing the server identity and trust keystores. You can also use it to manage the agent keystores. For more information, see "Managing Keys and Certificates with the Keystore Service" in *Oracle® Fusion Middleware Securing Applications with Oracle Platform Security Services* .

Table 2-3 lists the default values for the OPSS settings.

Table 2-3 Oracle Platform Security Services Settings

Setting	Value
Name of the application stripe created by the server	`VeridataSec`
Name of the identity keystore under the `VeridataSec` application stripe	`VeridataWebIdentityStore`
Name of the trust keystore under the `VeridataSec` application stripe	`VeridataWebTrustStore`

To configure two-way SSL by using the Oracle Platform Security Services keystore service:

1. For each agent, create an identity and trust keystore.
2. Update `VeridataWebIdentityStore` with the identity certificate of the server.
3. Update `VeridataWebTrustStore` with all agent certificates.
4. Update each agent truststore with the identity certificate of the server.
5. Export the agent keystore and truststore as JKS files and note the passwords.
6. Distribute the JKS files to corresponding agent machines.
7. Run the `configure_agent_ssl` tool to update the agent wallet with the keystore passwords.
8. For each agent, configure the `agent.properties` file to enable SSL.

To configure one-way SSL by using the Oracle Platform Security Services keystore service:

1. For each agent, create an identity keystore.
2. Update the `VeridataWebIdentityStore` with the identity certificate of the server.
3. Export the agent keystore and truststore as JKS files and write down the passwords.
4. Distribute the JKS files to corresponding agent machines.
5. Run the `configure_agent_ssl` tool to update the agent wallet with the keystore passwords.
6. For each agent, configure the `agent.properties` file to enable SSL.

Modifying the Oracle GoldenGate Veridata Agent Wallet

Before you start the agent in SSL mode, you must update the agent wallet with the identity and trust keystore passwords. Otherwise, the agent doesn't start.

To update the wallet:

1. Run the `configure_agent_ssl` script that is available in the agent home:

   AGENT_HOME`\configure_agent_ssl.sh` *AgentID*

   `AgentID` is the name of the agent properties file, without the `.properties` extension. The default value for `AgentID` is `agent`.
2. At the prompts enter the entry or unlock password for the identity and trust keystores for the agent.

Securing the Oracle GoldenGate Veridata Files

This topic describes how to secure your business data and control access to the Oracle GoldenGate Veridata installation directories and user interface.

Controlling Access to the Installation Directories

Standard operating system permissions apply to the programs, files, and directories in the server, agent, and web user interface installation directories. You should adjust the permissions for these objects based on your business security rules.

Securing Files That Contain Business Data

The server creates data files that contain sensitive application data. By default, these files reside in `DOMAIN_HOME/veridata/reports`. The subdirectories contain files that may reflect business data.

The following types of files contain sensitive data:

- Comparison report (`rpt` subdirectories)
- Out-of-sync report (`oosxml` and `oos` subdirectories)

These files inherit the same file permissions as those of the user that runs the server installation program. Do not change the permissions; if you do, then Oracle GoldenGate Veridata may be unable to maintain them. Keep these files just as secure as you would keep your business data. Users of the Oracle GoldenGate Veridata web user interface do not require access to these files because they see the same information through the client interface. You can encrypt the contents of all report files.

Securing Access to Oracle GoldenGate Veridata by Defining User Roles

You assign security roles to control user access to the software functions, some of which expose selected data values from the database.

Oracle GoldenGate Veridata	Type	Description
`veridataAdministrator`	Type-A	The administrator role is the highest level security role. This role can perform all functions that configure, execute, and monitor Oracle GoldenGate Veridata.
`veridataPowerUser`	Type-A	The power user role is the second highest role. This role can perform all functions that configure, execute, and monitor Oracle GoldenGate Veridata from the web user interface. It cannot perform any configuration functions for the server.
`veridataReportViewer`	Type-B	The report viewer role cannot perform functions that configure Oracle GoldenGate Veridata or execute jobs. This role can only view configuration and job information, and view comparison reports.
`veridataDetailReportViewer`	Type-B	The detail report viewer role cannot perform any functions that configure Oracle GoldenGate Veridata or execute jobs. This role can only view configuration and job information. It can also view comparison reports and out-of-sync report information through the web user interface or at the file level.
`veridataRepairOperator`	Additional	The repair operator role can use the Repair feature in Oracle GoldenGate Veridata.
`veridataCommandLineUser`	Additional	The command-line user role provides access to the Oracle GoldenGate Veridata command-line tools, `vericom`, and the Oracle GoldenGate Veridata import and export utilities.

These roles are categorized into as follows:

- Type A and Type B: By default, Type A and Type B users are not given any privileges of the additional user roles. Assign additional roles to users of these types.
- Additional: WebLogic Administrators can assign these Additional roles to Type A users to perform the required Oracle GoldenGate Veridata functions.

Security is controlled through the Oracle WebLogic Server Administration Console, where a user with the administrator role can:

- Create a user and assign it a security role.
- Create user groups and assign them security roles. Users can be added to these groups without being given a security role. A user inherits the role of its group.
- Create a user and assign it a security role, and then add that user to a group. The user inherits the role of its group and keeps its individual role.

Setting Up Active Directory Users on Oracle GoldenGate Veridata

To set up Active Directory Users on Oracle GoldenGate Veridata:

1. In the Oracle WebLogic Server Administration Console, navigate to **Security Realms**, then select **myrealm** , and then click **Roles and Policies**.
2. Select the **Realm Roles** subtab.
3. Expand **Global Roles** and then click on **Roles** link from the table.

4. Click **New** to create the following Roles:
   a. `ExtAdministrator`
   b. `ExtPowerUser`
   c. `ExtDetailReportViewer`
   d. `ExtReportViewer`
   e. `ExtReportViewer`

Oracle GoldenGate Veridata	Type	Description
`ExtAdministrator`	Type-A	The `ExtAdministrator` role is the highest-level security role in Oracle GoldenGate Veridata. This role can perform all of the functions that configure, execute, and monitor Oracle GoldenGate Veridata.
`ExtPowerUser`	Type-A	The `ExtPowerUser` role is the second-highest role in Oracle GoldenGate Veridata. This role can perform all of the functions that configure, execute, and monitor Oracle GoldenGate Veridata from the Oracle GoldenGate Veridata Web User Interface, but this role cannot perform any configuration functions for the Oracle GoldenGate Veridata Server.
`ExtReportViewer`	Type-B	The `ExtReportViewer` role cannot perform functions that configure Oracle GoldenGate Veridata or execute jobs. This role can only view configuration and job information, and view comparison reports.
`ExtDetailReportViewer`	Type-B	The `ExtDetailReportViewer` role cannot perform any functions that configure Oracle GoldenGate Veridata or execute jobs. This role can only view configuration and job information, and view comparison reports and out-of-sync report information through the Oracle Oracle GoldenGate Veridata web user interface or at the file level.
`ExtRepairOperator`	Additional	The `ExtRepairOperator` role can use the Repair feature in Oracle GoldenGate Veridata.

To open the Oracle WebLogic Server Administration Console:

1. In your browser, connect to the Oracle WebLogic Server Administration Console by entering the following address:

   `http://weblogic_admin_server_hostname:admin_server_port/console`

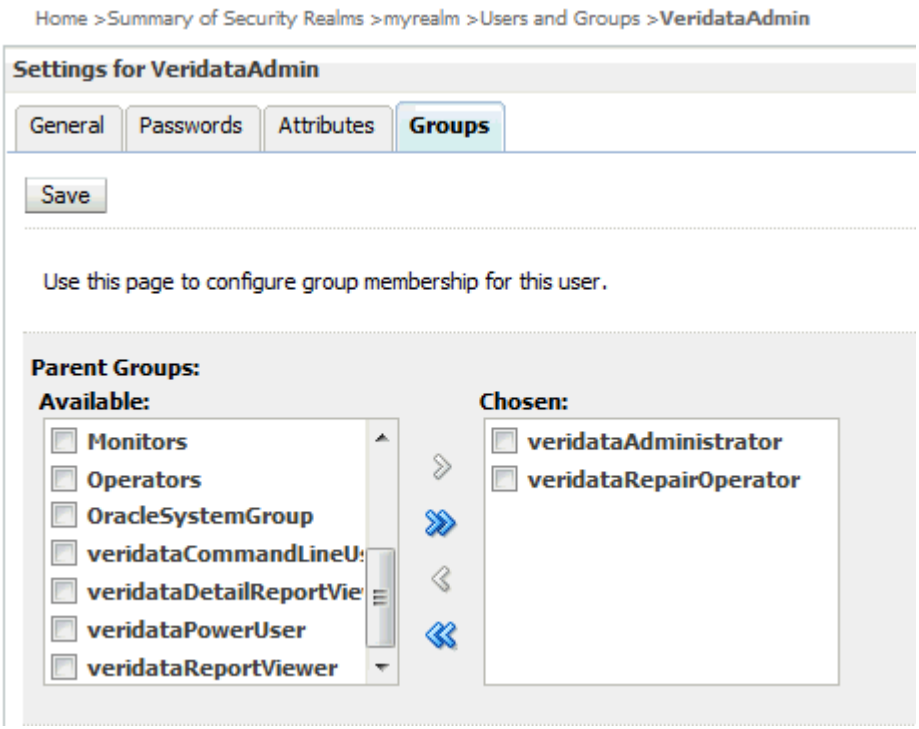
   weblogic_admin_server_hostname is the name or IP address of the system where the server and web components are hosted, and *admin_server_port* is the port number assigned to the server (default is 7001).

2. Log on as an Oracle GoldenGate Veridata administrator user. A default administrator user was created when the Oracle GoldenGate Veridata domain was set up.

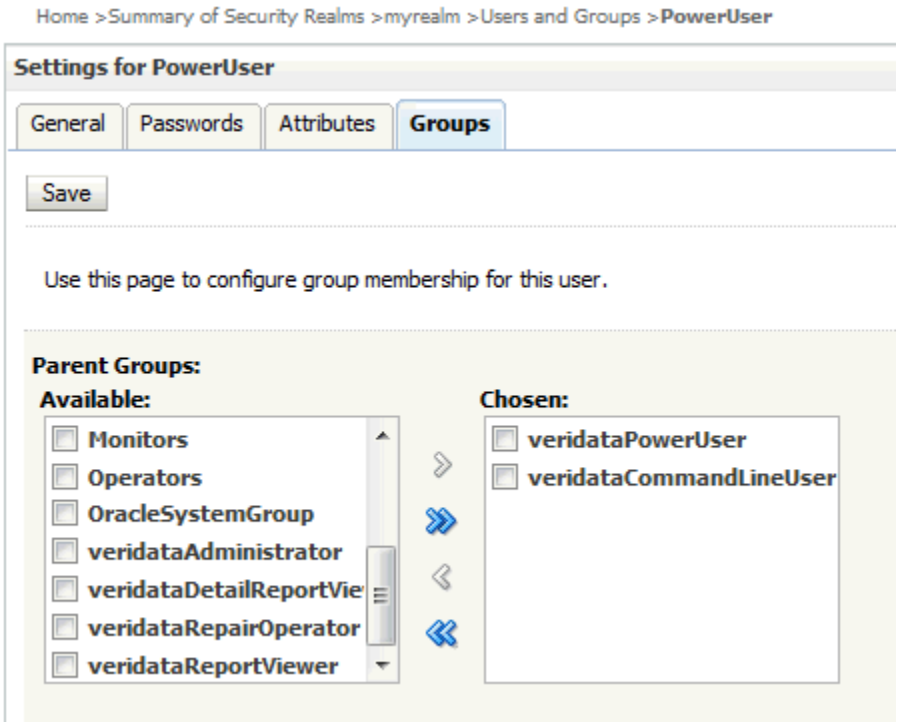
To create or edit a user:

1. In the left pane of the Oracle WebLogic Server Administration Console, select **Security Realms**.
2. On the Summary of Security Realms page select the name of the security realm.
3. On the Settings for Veridata Security realm page, select **Users and Groups,** and then select **Users**. The User table displays the names of all users defined in the Authentication provider.
4. Select an existing user and edit the settings or click **New** to create a user and enter the user's properties.
5. To assign a role to the user, click **Groups** on the Settings for *user_name* page. Select appropriate roles for the user.

   For example, an administrator, **VeridataAdmin**, can be given the privileges shown in the figure.



For example, a Veridata power user, **PowerUser**, can be given the privileges shown in the next figure.



6. Click **Save** .

Changing Database Schema Passwords

You can change database schema passwords when a schema password expires, an account is locked, or a password change is necessary. This topic applies to all database schemas that are prefixed with 'OGG', such as `OGG_IAU, OGG_IAU_APPEND`, or `OGG_IAU_VIEWER`.

T change a database schema password:

1. Stop your Oracle GoldenGate Veridata Server.

   ```
   DOMAIN_HOME/veridata/veridata/bin/veridataServer.sh stop
   ```

2. Stop your Oracle WebLogic Server.

   ```
   DOMAIN_HOME//veridata/bin/stopWebLogic.sh
   ```

3. Start the Oracle WebLogic Scripting Tool, `wlst.sh`. For example:

   ```
   /home/oracle/Oracle/Middleware/Oracle_Home/oracle_common/common/bin/wlst.sh
   ```

4. Modify the database schema password as in this example:

   ```
   modifyBootStrapCredential(jpsConfigFile='/home/oracle/wls_domains/veridata/
   config/fmwconfig/jps-config.xml',username='OGG_OPSS',password='welcome123')
   ```

   `username=` is your database schema name and `password=` is the new password.

5. Exit the scripting tool by using **`exit()`**.

6. Log in to the metadata repository database and change the schema passwords and unlock the schema.

   The following example that unlocks the `OGG_OPSS` schema:

   ```
   alter user OGG_OPSS identified by welcome123;
   alter user OGG_IAU identified by welcome123;
   alter user OGG_IAU_APPEND identified by welcome123;
   alter user OGG_IAU_VIEWER identified by welcome123;
   alter user OGG_STB identified by welcome123;
   alter user OGG_VERIDATA identified by welcome123;
   alter user OGG_OPSS account unlock;
   ```

7. Start the Oracle WebLogic Server Configuration Wizard. For example:

   ```
   /home/oracle/Oracle/Middleware/Oracle_Home/oracle_common/common/bin/config.sh
   ```

8. Select **Update an Existing Domain**, and then click **Next**.
9. Select **JDBC Component Schema**, and then enter the new database schema passwords.
10. Click **Next** until you reach the **Configuration Summary** screen, and then click **Update**.
11. To save your schema password changes, click **Next** and then **Finish.**
12. Start Oracle WebLogic Server.

    ```
    DOMAIN_HOME//veridata/bin/startWebLogic.sh
    ```

13. Start your Oracle GoldenGate Veridata Server.

    ```
    DOMAIN_HOME/veridata/veridata/bin/veridataServer.sh start
    ```

Encrypting Report Files

You can encrypt the comparison report files (`.rpt`, `.oos`, `.oosxml`) in Oracle GoldenGate Veridata. The following sections explain report encryption:

- Enabling Report Encryption
- Displaying Encrypted Files with the reportutil Utility

Enabling Report Encryption

The encryption is controlled by the following parameters in the `veridata.cfg` configuration file:

- `server.encryption`
- `server.encryption.bits`

To enable encryption, set `server.encryption` to `true`.

Encryption of report files uses Advanced Encryption Standard (AES) encryption, and the default encryption strength is 128 bits. You can increase the encryption strength to 192 or 256 bits by editing the value of the `server.encryption.bits` parameter in `veridata.cfg`. However, for encryption strength greater than 128, you must use a JRE installed with the Unlimited Strength Cryptography Extension.

For more information about these parameters, see "Parameters for Report File Encryption".

Encrypted report files have the following extensions in the file names:

- `.xrpt` : Encrypted comparison or repair report file
- `.xoos`: Encrypted binary out-of-sync file
- `.xoosxml`: Encrypted out-of-sync XML file
- `.xNNN`: Encrypted out-of-sync XML chunk file (`NNN` is a decimal number)

Displaying Encrypted Files with the reportutil Utility

When report encryption is enabled, all report files are encrypted by using an encryption key, which is initially a large random value. If necessary, then you can change the encryption key.

Before you can read encrypted files, you must decrypt them. The Oracle GoldenGate Veridata web user interface automatically decrypts files before displaying them. Use the `reportutil.sh/.bat` utility to display the encrypted contents. This utility is located in the *`VERIDATA_DOMAIN_HOME`*`\veridata\bin` directory.

To display encrypted files, run the `reportutil.sh/.bat` utility located in the *`VERIDATA_DOMAIN_HOME`*`\veridata\bin` directory.

```
reportutil   [-wlport port ] -wluser weblogic_user { options }
```

`wlport` is the server port number (the default port is 8830), and `wluser` is the server user name.

The valid `options` are:

- `-version`, `-v`: Displays the current version
- `-help`: Displays the help message
- `-r`: Rolls report encryption
- `-f` *`filename`* [`-d` *`directory`*]: Decrypts and prints the report file to the specified file if a directory is specified by the `-d` option. Otherwise, the command prints the decrypted file to the standard output.

The user running the `reportutil` utility must be in the appropriate user group to perform the operations:

- `-r`, `-f`: Allowed only if the user is a member of the `veridataCommandLineUser` group.
- `-r`: Allowed if the user is a member of the `veridataAdministrator` group.
- `-f`: Allowed if the user is a member of the `veridataAdministrator` group, or a member of `veridataPowerUser` group, the `veridataPowerUser` group, or the `veridataDetailReportViewer` group.

For more information about the user roles, see Securing Access to Oracle GoldenGate Veridata by Defining User Roles.

3

Managing C-Agent, Manager, and Java Components

In this chapter, you learn how to manage C-agent, manager, and Java-based components, including controlling logging levels and connecting to the web user interface.

This topic includes the following sections:

Starting and Stopping the C-Agent and the Oracle GoldenGate Veridata Manager

When the Oracle GoldenGate Veridata (server) initiates comparisons, the C-agent starts automatically. However, for Oracle GoldenGate Veridata Agent (agent) to function correctly, the following must be running:

- The database to which the agent is linked.
- The Manager process for the C-agent.

Although the agent process itself is automatic, you can stop the Manager process that controls the agent. Stopping Manager prevents the server from being able to start a new agent process, but it does not stop agents that are already running.

To control the C-agent Manager on all platforms

1. From the agent installation location, run the Oracle GoldenGate software command-line interface.
2. Stop or start the Manager.

```
START MANAGER

STOP MANAGER
```

Starting and Stopping the Java-Based Components

The server and web user interface components are Java programs. The agent component is also available as a Java program for all platforms except NonStop.

> **Note:**
>
> Before starting the server and web processes, start the repository database.

To start and stop the agent component:

1. Switch to its installation directory.
2. Start or stop the agent.

   UNIX or LINUX
   `agent.sh {start | run}` OR `agent.sh stop`

   Windows
   `agent.bat {start | run}` OR `agent.bat stop`

   - In these commands:
   - `run` starts the agent in the same command window where it is launched.
   - `start` starts the agent in a separate command window.

   > **Note:**
   >
   > The `run` option is useful for diagnosing errors that happen during the startup process before the agent error logging is configured. When the `run` option is used, messages written to `stdout` and `stderr` appear in the command window. The agent normally logs its messages to the log file, so only operating system messages and logging system errors are written to `stderr`. When the start option is used, messages written to `stdout` and `stderr` are discarded.

Configure the host to start and stop the processes automatically. Contact your system administrator if you need assistance.

Reloading Logging Information

You can reload logging information from the `AGENT_ORACLE_HOME/config/odl.xml` configuration file to a running agent.

To reload logging information, start the agent and run the reloadLog commands shown in the following table:

UNIX and Linux	Windows
`agent.sh reloadLog`	`agent.bat reloadLog`

Controlling Logging Levels for Oracle GoldenGate Veridata Agent

To control the agent logging levels so that they log more information for debugging purposes, update the `AGENT_ORACLE_HOME/config/odl.xml`configuration file. Relevant comments are provided in the `odl.xml` file.

Connecting to the Oracle GoldenGate Veridata Web Interface

To connect to the Oracle GoldenGate Veridata web user interface:

1. In a web browser, enter the following address: `http://`*`hostname`*`:`*`port`*`/veridata`.

   In this example, `hostname` is the name of the system where the server is installed and `port` is the port number where it is running (default is 8830). Use `localhost` as the host name if you are connecting on the system that is local to the server installation.

   Examples:

   ```
   http://localhost:8830/veridata
   http://sysa:8830/veridata
   ```

2. On the Oracle GoldenGate Veridata Web login page, enter your user name and password. For full instructions on using the Oracle GoldenGate Veridata web user interface, see the online help.

4

Managing Identities and Comparing Data

The `Vericom` command-line interface provides a tool for you to manage identities in the credential store and run comparisons.

This chapter includes the following sections:

This chapter includes the following sections:

Overview of the Vericom Tool

You can use the `vericom` tool to execute certain comparison tasks from the command shell of the operating system. The `vericom` tool runs theOracle GoldenGate Veridata command-line interface and enables you to handle these activities with automated programs. You can easily use the `vericom` tool without specifying the actual user name and password.

You can:

- Run an entire job or a specific compare pair of a job

> **Note:**
>
> You cannot run a group individually.

- Set tracing (only under guidance of an Oracle Support analyst)

For specific compare pairs, you can:

- Review previous out-of-sync results
- Generate out-of-sync XML from the previous run
- Override the same profile and row partition settings that are possible from the web user interface

You can also run comparisons from the Oracle GoldenGate Veridata web user interface. This interface provides greater control for configuring the objects to be compared and for controlling runtime parameter settings.

Managing Identities in a Credential Store

This section shows you how to use a credential store to maintain encrypted database passwords and user IDs and associate them with an alias. It is the alias, not the actual user ID or password, that is specified in a command or parameter file . No user input of an encryption key is required. The credential store is implemented as an Auto Login wallet within the Oracle Credential Store Framework.

Credential Store contains the following topics:

- Adding a Credential Store
- Deleting a Credential Store

Alias contains the following topics:

- Creating an Alias
- Using the Alias
- Display Alias
- Updating the Alias
- Deleting the Alias

Adding a Credential Store

The `–addCredentialStore` argument does not accept any input. The default location for credential store is `<Domain_home>/veridata/dircrd`. You can change this by specifying a directory in the `veridata.cfg` file under the `credential.store.location` property. The default value of the `credential.store.location` property is `veridata/dircrd` which is relative to the domain location. You can create your own credential store location and change this property before running `vericom`. The credential store wallet is created with only read and write permissions (-rw-------).

Example 4-1 Example

```
vericom.sh -wluser username -wlport veridata server port –addCredentialStore.
```

Deleting a Credential Store

The `–deleteCredentialStore` argument does not accept any input. This option deletes the credential store. The location for the credential store is `credential.store.location` in the `veridata.cfg` file. The credential store wallet and its contents are permanently deleted.

Example 4-2 Example

```
vericom.sh -wluser username -wlport veridata server port –deleteCredentialStore
```

Creating an Alias

The `-createAlias` argument accepts zero or one as input. If the input is provided, then it is used as an alias name. If it is not provided, then the user name provided in the `wluser` argument is used as an alias. This alias is used in place of the user name and password, and you do not have to provide an actual user name and password.

Example 4-3 Example

```
vericom.sh -wluser username-wlport veridata server port –createAlias aliasname (optional)
```

Using the Alias

You must use the `-wluserAlias` argument with the alias that you created with `-createAlias`. With this option, you are not prompted for a password. You should not use the `-wluserAlias` argument with `-wluserAlias`. If the alias does not exist in the wallet, then an error is returned.

Example 4-4 Example

```
vericom.sh -wluserAlias alias-name -wlport veridata server port –job job name
```

Display Alias

Use the `-displayAlias` argument to list all aliases and user names in the wallet. The password is not displayed.

Example 4-5 Example

```
vericom.sh –wluser username -wlPort veridata server port -displayAlias
```

Updating the Alias

Use the `-updateAlias` argument to update the password for the valid user.

Example 4-6 Example

```
vericom.sh -wluser username -wlport veridata server port –updateAlias alias-name (optional)
```

Deleting the Alias

The `-deleteAlias` argument used to delete the alias. Along with the alias, the corresponding user credentials are also removed if, no other alias is referring to the same user.

Example 4-7 Example

```
vericom.sh -wluser username -wlport veridata server port –deleteAlias aliasname (optional)
```

Running the Vericom Tool

Anyone who has operating system permissions can run the `vericom` tool.

1. On the system where Oracle GoldenGate Veridata is installed, run the operating system's command shell.
2. Navigate to the *VERIDATA_DOMAIN_HOME*`/veridata/bin` directory.
3. Run the `vericom` tool: `vericom{.bat|.sh}` *required_parameter* `[`*optional_parameter*.

Required Parameters

Enter one of the following required options; otherwise, an error is returned.

```
[-wlport port ] |
-wluser user_name |
-help |
-helprun |
[-version | -v] |
[-job | -j] job |
```

In this example, the `-wluser` option specifies the Oracle GoldenGate Veridata Server (server) user name that is needed to connect to the server. This user should have the `veridataCommandLineUser` privilege to access and execute command-line operations. The user should also have the `veridataAdministrator` or `veridataPowerUser` privilege to successfully run jobs and use the import and export utilities.

See Securing Access to Oracle GoldenGate Veridata by Defining User Roles.

The `-version`, `-v`, `-help`, or `-helprun` options, they take precedence over any other option specified.

Optional Parameters

These are the optional parameters:

```
[ -g group -c compare_pair ]
[ -nw ]
[ -repair | -norepair]
[ -rP profile ]
[ -rR ]
| -rO ]
[ -rN threads ]
[ -rD seconds ]
[ -rC | +rC ]
[ -rOb | -rOx | -rO2 | -rO0 ]
[ -rOs records ]
[ -rTi ]
[ -rTc ]
[ -rTs trace_number ]
[ -pS source_partition_name |
    -pSq source_sql_predicate |
    -pSA1 source_ascii_start_key |
    -pSA2 source_ascii_end_key |
    -pSH1 source_hex_start_key |
    -pSH2 source_hex_end_key ]
[ -pT target_partition_name |
    -pTq target_sql_predicate |
    -pTA1 target_ascii_start_key |
    -pTA2 target_ascii_end_key |
    -pTH1 target_hex_start_key |
    -pTH2 target_hex_end_key ]
[ -pq sql_predicate ]
[ -rd0 | -rdN run_ID ]
[ -wp ]
-addCredentialStore
       Create a new Credential Store at location defined in the veridata.cfg file
  -deleteCredentialStore
       Delete the Credential Store
  -createAlias
```

```
      Create an alias for user provided in the wluser argument
  -updateAlias
      Update the user name and password for the alias with user provided in the
wluser argument
  -deleteAlias
      Delete the alias
  -displayAlias
      Display the alias stored in credential store
  -wlUserAlias
      Use the alias in place of wluser
```

Table 4-1 Vericom Runtime Arguments

Argument	Description
`-wluser`	Specifies the server user name that authenticates and connects to the server.
`-wlport`	Specifies the server port number.
`-help`	Displays the `vericom` syntax components and their descriptions.
`-helprun`	Displays run-related syntax components and their descriptions.
`{-version | -v}`	Displays the version of the Oracle GoldenGate Veridata command-line interface that is being used.
`{-job | -j} job`	Specifies the job to be run. For *`job`*, specify the name that was assigned when the job was created in Oracle GoldenGate Veridata Web.
`-g group` `-c compare_pair`	Specifies a group and compare pair. For *`group`* and *`compare_pair`*, specify the names that were assigned when these objects were created in Oracle GoldenGate Veridata Web. • If `-g` and `-c` are used, `-j` must also be used.
`-nw`	Directs `vericom` not to wait for the job to finish before returning the prompt. Instead, `vericom` returns the prompt immediately after starting a job.
`-repair | -norepair`	Specifies whether to repair after a comparison is completed and confirms that out-of-sync data exists
`-rP profile`	Overrides the defined for a job. For profile that is defined for a job. For *`profile`*, specify the name that was assigned when the profile was created in Oracle GoldenGate Veridata Web. If `-rP` is used, then `-j` must be used.
`-rR`	A run override option. Compares only those rows that were out-of-sync in the previous run, based on the information that is stored in the out-of-sync file. The results identify which rows were brought back into synchronization by replication or another method. • Don't use `-rR` and `-r0` in the same run.

Table 4-1 (Cont.) Vericom Runtime Arguments

Argument	Description
`-rO`	A run override option. Generates an `OOSXML` file that is based on the out-of-sync file from the previous run. It generates XML for every row that is in the file. You can use the XML to view the out-of-sync information in an XML editor or for other purposes. • You must use `-rO` with `-j`. • Don't use `-rR` and `-rO` in the same run.
`-rN` *threads*	Specifies the number of concurrent comparison threads to use. You can use as many threads as there are processors on the server system. This option overrides the default job profile and has no effect if a job is not run with `-j` or if just one comparison is run by using `-j` with `-g` and `-c`.
`-rD` *seconds*	Delays the confirmation step by the specified number of seconds to account for replication lag. Delaying the confirmation step reduces the number of false out-of-sync results that occur because an updated source value was not replicated fast enough. This option overrides the default job profile and has no effect if you use the `-rR` option.
`-rC | +rC`	Controls whether or not the confirmation step (confirm OOS) is performed in the job. • `-rC` skips the confirmation step. You can skip the confirmation step if activity on the source tables is stopped or if replication doesn't continuously update the target tables. • `+rC` includes the confirmation step. These options override the default job profile and are mutually exclusive. They have no effect unless you use `-j`.
`-rOb | -rOx | -rO2 | -rO0`	Controls the kind of file that is produced for the out-of-sync report. • `-rOb` generates binary format that is compatible with the Oracle GoldenGate Veridata Web browser. • `-rOx` generates output in XML. • `-rO2` generates both binary and XML output. • `-rO0` suppresses out-of-sync output. These options override the default job profile and are mutually exclusive. They have no effect if you use `-rR`.
`-rOs` *records*	Limits the number of out-of-sync rows that are written to a chunk of the `OOSXML` file. Writing the file in chunks prevents it from becoming too large for the system to manage and allows periodic archiving or purging. The current file is closed when the specified number of rows is written, and a new file is opened. This option overrides the default job profile and has no effect if you use `-rR`.
`-rTi`	Turns on tracing of Oracle GoldenGate Veridata Agent (agent) for the initial comparison step. Don't use it without the guidance of an Oracle support analyst.

Table 4-1 (Cont.) Vericom Runtime Arguments

Argument	Description
`-rTc`	Turns on tracing of the agent for the confirmation step. Don't use it without the guidance of an Oracle support analyst.
`-rTs` *`trace_number`*	Turns on tracing for the server. *`trace_number`* is a bitmask of server execution trace flags. A higher level of trace flags results in more detailed trace data.
`-pS` *`source_partition_name`* | `-pSq` *`source_sql_predicate`* | `-pSA1` *`source_ascii_start_key`* | `-pSA2` *`source_ascii_end_key`* | `-pSH1` *`source_hex_start_key`* | `-pSH2` *`source_hex_end_key`*	Runs the comparison using an existing source row partition or using an override partition that is defined by partition criteria. These options are mutually exclusive. They are valid only if comparing one compare pair (`-j` with `-g` and `-c`) and are ignored otherwise. **`-pS` *`source_partition_name`*** Specifies an existing source partition that is already defined and stored in the repository. The partition name is not validated and is passed directly to the server. An error is returned if the specified partition does not exist. **`-pSq` *`source_sql_predicate`*** Specifies a SQL predicate that defines a partition to override an existing source partition for a SQL table. The predicate is the conditional statement that follows the `WHERE` keyword, for example: `LAST_NAME BETWEEN "A" AND "M"`. Do not include the `WHERE` keyword. It will be added automatically at runtime. If the predicate contains multiple words, it must be enclosed within quotes to make it a single command argument. The type of quote is dependent on the command shell or interpreter that is being used. If the predicate contains special characters (such as $, *, < in sh/csh or %, < in Windows), then they must be properly escaped for that shell or interpreter. **`-pSA1` *`source_ascii_start_key`*** Specifies an ASCII key as the starting key value of a partition that overrides an existing source partition for an Enscribe file. **`-pSA2` *`source_ascii_end_key`*** Specifies an ASCII key as the ending key value of a partition that overrides an existing source partition for an Enscribe file. **`-pSH1` *`source_hex_start_key`*** Specifies a hexadecimal key as the starting key value of a partition that overrides an existing source partition for an Enscribe file. **`-pSH2` *`source_hex_end_key`*** Specifies a hexadecimal key as the ending key value of a partition that overrides an existing source partition for an Enscribe file.

Table 4-1 (Cont.) Vericom Runtime Arguments

Argument	Description

`-pT target_partition_name|`
`-pTq target_sql_predicate |`
`-pTA1 target_ascii_start_key |`
`-pTA2 target_ascii_end_key |`
`-pTH1 target_hex_start_key |`
`-pTH2 target_hex_end_key`
	These options specify target partitions and have the same rules as the corresponding options that specify source partitions.

`-pq sql_predicate`	Specifies a SQL predicate to be used for both the source and target SQL tables, as an override to existing partitions. This option has the same rules as `-pSq source_sql_predicate` and `-pTq target_sql_predicate`.
`-rd0 | -rdN run_ID`	Controls delta processing for a compare pair. • `-rd0` disables delta processing for this run. All rows are compared. • `-rdN run_ID` enables delta processing by using a previous job run as the basis for the delta. For `run_ID` use the number from the `Run ID` line at the beginning of the job comparison report. Vericom does not validate the run ID that is supplied. To use these options, you must specify a compare pair with: • `-j` • `-g` • `-c`
`-wp seconds`	Waits for a job to complete. The client also polls of the status of job submitted to the server at the specified interval (in seconds). • The argument accepts no data or a single data as input. If the input is provided, then it is used as an alias name. If no input is provided, then the user name provided in the wluser argument is taken as an alias. This alias is used in place of the user name and password. You do not have to provide the actual user name and password. `-wp` must be used with `-job | -j`.
`-addCredentialStore`	This argument does not accept any input. The default location for credential store is <*Domain_home*>/*veridata/dircrd*. You can change the directory by specifying it in the *veridata.cfg* file under property *credential.store.location*. The default value of *credential.store.location* property is *veridata/dircrd* which is relative to the domain location. You can create your own credential store location and change this property before running vericom. The credential store wallet is created with only read and write permission *(-rw-------)*.

Table 4-1 (Cont.) Vericom Runtime Arguments

Argument	Description
`-deleteCredentialStore`	This argument does not accept any input. Use this option to delete the credential store. The location for credential store is the value of property *credential.store.location* defined in `veridata.cfg` file. The credential store wallet and its contents will be permanently deleted.
`-createAlias`	The argument accepts no data or a single data as input. If the input is provided, then it is used as an alias name. If no input is provided, then the user name provided in the wluser argument is taken as an alias. This alias is used in place of the user name and password. You do not have to provide the actual user name and password.
`-wluserAlias`	This argument is used with an alias that is created by using `-createAlias` in place of user name. You aren't prompted for a password. You aren't required to use this option with `-wluser` argument. If the alias does not exist in the wallet then an error is returned.
`-displayAlias`	Use this argument to list all aliases in the wallet. Only the alias and user name are displayed.
`-updateAlias`	Use this argument is used to update your password.
`-deleteAlias`	Use this argument to delete the alias that you created in the wallet.

Vericom Exit Statuses

The `vericom` command-line tool exits with one of the following statuses. The examples are for a UNIX or Linux system.

Vericom exits with one of the following statuses. This examples shown are for a UNIX or Linux system.

Table 4-2 Vericom Exit Status

Status	Description
`0`	The command executed successfully. If a job was run, then all rows are in-sync. If you specified `-nw`, then the exit status is 0 if the job started successfully.
`1`	Invalid `vericom` syntax was used. For example, the following are invalid: `vericom.sh -helptun` (A typographical error occurred.) `vericom.sh -j -g group1` (The name of the job is missing.)

Table 4-2 (Cont.) Vericom Exit Status

Status	Description

3	Provides more granularity for input errors that involve comparison flags. For example, the following mistakes cause this error:
	`vericom.sh –j job1 –c address=address`
	In the preceding example, the `-g` *`group`* input is missing. It is required with `-j` if `-c` is used.
	`vericom.sh –j job1 –g group1 –rd0`
	In the preceding example, the `-rd0` flag requires `–c` because delta processing applies at the compare pair level.

4	The job ran successfully, but the comparison of some rows are not in sync.
5	There was a communication error with the server.

Vericom Output Examples

To view the results of a comparison that you run with the vericom tool, you can use the Oracle GoldenGate Veridata web user interface to view the comparison report. You can also view the output that is returned by the tool to the terminal. If a run finishes successfully, statistics for the job are displayed.

See Viewing Comparison Results.

The following examples use the `TestJob` job:

Example 1
This example shows a run on a Windows system without specifying `-w`. The process exits with status 0, and finished job statistics are not displayed.

```
VERIDATA_DOMAIN_HOME\veridata\bin\vericom.bat -wluser veridata -wlport 8830 -j
TestJob
Connecting to: localhost:9177
Run ID: (2256, 0, 0)
C:\veridata\server\bin> if errorlevel 0 echo EXITED 0 STATUS
EXITED 0 STATUS
```

Example 2
This example shows a run of the `TestJob` with `-w` specified. The process exits with status 4 because one of the compare pairs had a validation error. Finished job statistics are displayed.

```
VERIDATA_DOMAIN_HOME\veridata\bin\vericom.bat -wluser veridata -wlport 8830 -j
TestJob -w
Connecting to: localhost:9177
Run ID: (2257, 0, 0)
Job Start Time: 2008-03-21 22:48:05
Job Stop Time: 2008-03-21 22:48:20
Job Report Filename: C:\testjunit\rpt\TestJob\00002257\TestJob.rpt
Number of Compare Pairs: 3
Number of Compare Pairs With Errors: 1
```

```
Number of Compare Pairs With OOS: 1
Number of Compare Pairs With No OOS: 1
Number of Compare Pairs Cancelled: 0
Job Completion Status: WITH ERRORS
C:\veridata\server\bin> if errorlevel 4 echo EXITED 4 STATUS
EXITED 4 STATUS
```

Example 3

This example shows a run of the `TABLE9=TABLE9` in job `TestJob` with `-w` specified. The process exits with status 0 because the tables are in sync. Finished job statistics are displayed.

```
VERIDATA_DOMAIN_HOME\veridata\bin\vericom.bat -wluser veridata -wlport 8830 -j
TestJob -g TestGroup -c TABLE9=TABLE9 -w
Connecting to: localhost:9177
Run ID: (2258, 0, 0)
Job Start Time: 2008-03-21 22:51:08
Job Stop Time: 2008-03-21 22:51:11
Job Report Filename: C:\veridata\data\rpt\TestJob\00002258\TestJob.rpt
Number of Compare Pairs: 1
Number of Compare Pairs With Errors: 0
Number of Compare Pairs With OOS: 0
Number of Compare Pairs With No OOS: 1
Number of Compare Pairs Cancelled: 0
Compare Pair Report Filename: C:\veridata\data\rpt\TestJob\00002258\TestGroup\CP_
TABLE9=TABLE9.rpt
Number of Rows Compared: 21
Number of Rows In Sync: 21
Number of Rows With Errors: 0
Number of Rows Out Of Sync: 0
Number of Inserts Out Of Sync: 0
Number of Deletes Out Of Sync: 0
Number of Updates Out Of Sync: 0
Compare Pair OOSXML Directory: C:\veridata\data\oosxml\TestJob\00002258\TestGroup
Compare Pair OOSXML Filename:
Job Completion Status: IN SYNC
C:\veridata\server\bin> if errorlevel 0 echo EXITED 0 STATUS
EXITED 0 STATUS
```

On UNIX systems, the exit status is in the `'$?'` special variable if you use the SH or KSH shells. If you use the CSH shell, then the exit status is in the `'$status'` special variable.

5

Using the Veridata Import and Export Utilities

In addition to using the Oracle GoldenGate Veridata Web User Interface, you can use the import and export utilities, provided with the Veridata installation, to define portions of your configuration.
This chapter includes the following sections:

Introduction to the Import and Export Utilities

Using the import and export utilities, you can create XML documents that are used to configure Oracle GoldenGate Veridata. The DTD (Document Type Definition) that governs these XML documents is stored in the `ORACLE_HOME/veridata/clilib/lib/veridata-scripting.jar` file.

The import utility allows you to configure database connections, comparison groups including compare pairs, comparison jobs, and profiles. It takes an XML document as input then creates comparison objects in Veridata. Typically, the XML document matches the inputs on the configuration pages in the user interface.

The export utility helps you to either selectively or completely export the compare configuration data to an XML file. It can be used to export configurations from your current Veridata configuration or from other Veridata configurations using the `-repoUrl` option. Additionally, you can use it to export configurations between different Veridata repository types using the import functionality. For example, from a SQL Server configuration to an Oracle configuration.

You should have an understanding of basic XML and its rules.

These utilities provide the following advantages:

- It can reduce the time required to define repetitive tasks
- It allows you to create reusable configurations
- It can ensure that your test configuration mirrors the one you use for production

Supported Configurations

Oracle GoldenGate Veridata import and export utilities support configuring:

- Database connections
- Comparison groups (jobs, groups, and compare pairs)
- Profiles

Running the Import and Export Utilities

The import and export utilities run from the *DOMAIN_HOME*/veridata/bin directory of the Oracle GoldenGate Veridata installation location. The Windows programs are veridata_export.bat and veridata_import.bat; the UNIX and Linux scripts are veridata_export.sh and veridata_import.sh.

Using the Export Utility

The syntax for running the export utility is:

```
veridata_export[.sh | .bat] -export filePath -wluser commandlineUusername   [-
wlport portNo] [–jobs jobName | - groups groupName | -connections connName | -
profiles profileName | -all | -exportPassword] [[-repoUrl jdbc_url] [-u username>][-
schema schema_name][-vdtPath VERIDATA_PRODUCT_HOME]]
```

- -wlport: Represents the port for Veridata web server. The default value is 8830.
- -wluser: Specifies a user *commandlineUsername* with Veridata configuration privileges and command-line privileges.
- One of these optional operations can be requested at run time:
  - -jobs: Export all jobs, by name, including the associated groups, connections and profiles. You can specify one or more jobs by separating the names with a space, such as job1 job2 job3. If no job name is specified, all jobs with associated objects are exported.
  - -groups: Export all groups in the repository or add group names separated by a space, such as group1 group2 group3.
  - -connections: Export all connections in the repository or add connections separated by a space, such as conn1 conn2 conn3.
  - -profiles: Export all profiles in the repository or add profiles separated by a space, such as profile1 profile2 profile3.
  - -all: Export objects that are not part of any job. Takes precedence over all optional operations. This is the default when no other options are specified.
  - -exportPassword: Export the passwords for connections. By default, passwords for connections are not exported.
  - -repoUrl: Set the remote Veridata repository database JDBC URL for the export to use. You must set the -u option when using -repoUrl.

    For Oracle Database:

    ```
    jdbc:oracle:thin:@hostname/ip:port:SID
    ```

    or

    ```
    jdbc:oracle:thin:@hostname/ip:port/serviceName
    ```

    For SQL Server:

    ```
    jdbc:weblogic:sqlserver://hostname/ip:port;databaseName= databaseName
    ```

    The repoUrl may require double quotes.

  - `-u`: Set the remote Veridata database username from which the configuration export is requested. Use with the `-repoUrl` option.
  - `-schema`: Set the remote Veridata schema name from which the configuration export is requested.
  - `-vdtPath` *`veridataLocation`*: Set the Veridata domain location for the 12c release and later. For 11g release, it is the installation location.
  - `-help`: Provides command line syntax.

I f you want to export data from a Veridata repository database that is not part of your existing installation, you must provide the URL, username, and schema name. You will be prompted to enter the external Veridata repository password during run time. The Connection passwords are not exported by default, use the `-exportPassword` option for exporting passwords.

Using the Import Utility

The syntax for running the import utility is:

```
veridata_import[.sh | .bat] [-wlport portNo] -wluser commandlineUusername [-create
| -update | -delete | -replace] configuration.xml
```

- `-wlport`: Represents the port for Veridata web server. The default value is 8830.
- `-wluser`: Specifies a user *`commandlineUsername`* with Veridata configuration privileges and command-line privileges.
- One of these optional operations can be requested at run time:
  - `-create`: All items listed in the configuration are new. If any item in the list exists in the repository, nothing is added.
  - `-update`: New items are added to the repository and existing items are modified. Items existing in the repository and not listed in the configuration are deleted.
  - `-delete`: All named items that exist for the configuration are removed from the repository.
  - `-replace`: All items listed to be replaced in the configuration are replaced as specified.
- *`configuration.xml:`* The name of the import XML configuration file that you created to describe the configuration. This is a required option.

Processing the Configuration

The import utility first parses the *`configuration.xml`* file attempting to complete the entire file before aborting due to the errors. Any errors it finds are logged in the *`DOMAIN_HOME`*`/veridata/logs/veridata_import.log`. If it does not abort because of errors, it makes a second parsing pass, this time processing the configuration.

Matching Object Names

Database object names, such as catalogs, schema, tables, indexes, and columns will be matched according to these rules:

- The matching is case insensitive

- The hyphen (-) is considered a match to the underscore (_) to support matching Enscribe DDL and SQL columns
- Wildcard expressions for table names and source column names match against the exact name and against the uppercase version of the name.
- Quoted names for schemas and wildcards match everything within the quotations must be matched exactly. A wildcard character within quotes is treated as an ordinary character. An example of a quoted name as it would appear in the XML is:

  ```
  source-table="&quot;CHAR_TYPES&quot;*"
  ```

  This would match `CHAR_TYPES`, `CHAR_TYPES2`, and `CHAR_TYPES_NOTNULL`.
- Filters can either include or exclude schemas and tables. If include filters are used, at least one filter must be matched before a table can be included in a compare pair. If exclude filters are used, a table is excluded if it matches any exclude filter. Include filters can include a `COLFILTER` element that contains a list of columns to include or exclude. When a table matches a include filter, the include filter's `COLFILTER` is used to specify the columns for the generated compare pair. The schema and table name can use wildcards.

  For NonStop Enscribe files, file pattern filters are used. The file pattern is any valid NonStop file name pattern.
- A compare pair may have a column specification with the Boolean attribute "optional". When this attribute is true, the column is only included in the compare pair if the source table includes the specified source column.

Determining Key Columns

The key columns are selected in the following order:

1. Explicit key column definitions if they are available. In this case if `source-pkey` and `target-pkey` `compare-pair` element attributes are set it will generate an error.
2. Columns in the index specified by `source-pkey` and `target-pkey` attributes of the `compare-pair` element. The number of columns and all data types must match and the data types must be compatible.
3. Columns in the system-selected primary key.

Generating Compare Pairs

Compare pair generation has the following characteristics:

- Generating from wild cards works the same as the user interface generation except that regular expressions can be used.
- Compare pairs are processed in the order specified in the *`configuration.xml`* file
- The compare pairs generated by a single compare pair element are generated in alphabetical order of the source table name.
- When compare pairs are generated by more than one compare pair element, the first one will be used.

As a general rule, the order of the compare pair elements should be:

1. Compare pairs with specialized configuration requirements, such as user-defined keys.

2. Compare pairs that match general patterns.
3. Exclusions of compare pairs that would otherwise match general patterns.

Configuration File Element Reference

The configuration is defined by the top level `configuration` element and several nested elements. Most of these elements have attributes that define their characteristics, such as the `operation` attribute for the `configuration` element or the `port` attribute for the `connection` element.

The following is the high-level element hierarchy in the configuration XML file. For more information about an element and its attributes, click the element name in the hierarchy.

configuration
- connection
  - conn-properties
- group
  - description
  - filter
  - sql-partition
  - enscribe-key
  - compare-pair
    - enscribe-info
    - enscribe-key
    - sql-partition
    - column
    - excluded-column
    - delta-config
- job
- profile

Sample Configuration File provides a sample configuration file.

Profile Parameters provides a description of the profile parameters that you can use to configure your profiles.

configuration

The root element is `configuration`.

The following elements can be nested within the `configuration` element:

Table 5-1 configuration Elements

Elements	Description
`connection`	One or more Veridata database connection definitions.
`group`	One or more Veridata comparison group definitions.
`job`	One or more comparison job definitions.

Table 5-1 (Cont.) configuration Elements

Elements	Description
`profile`	One or more profile definitions.

The following attributes describe the `configuration` element:

Table 5-2 configuration Attributes

Attribute	Description
`validation`	Specifies the type of validation that is used for the configuration. The options are: "`required`" - All compare pairs must be successfully validated before any pairs are added to the repository. This is the *default* value. "`omit-failures`" - Successfully validated compare pairs are added to the repository and compare pairs that cannot be validated are ignored. "`none`" - Compare pairs are added to the repository without any validation. If this option is selected, the Oracle GoldenGate Veridata Web User Interface should be used to review and fix validation problems.
`operation`	Specifies how data is applied to the repository. The options are: "`create`" - All items listed in the configuration are new. If any item in the list exists in the repository, nothing is added. This can be used to prevent unintended modification to existing repository items. This is the *default* value. "`update`" - New items are added to the repository and existing items modified. Items existing in the repository and not listed in the configuration are deleted. "`delete`" - All named items in the configuration are removed from the repository. You can use a command line flag to override the value entered for this attribute.
`wildcard`	Specifies the pattern matching method that is used. The options are: "`ggs`" - Use the typical Oracle GoldenGate pattern using an asterisk (*). See the Oracle GoldenGate Veridata Web User Interface help for details on this type of matching. This is the default value. "`regex`" - Use regular expressions for matching.

Example

The following example adds compare pairs that can be validated and ignores those that cannot; uses regular expressions for wild carding; and uses the `"create"` default to adds all items as new items, adding nothing if any item already exists.

```
<configuration validation="omit-failures" wildcard="regex">
.
.
```

```
.
</configuration>
```

column

The `column` element defines a set of columns to be included or excluded from the compare pair. The `column` element has no nested elements or text data.

The following attributes describe the `column` element:

Table 5-3 column attributes

Attribute	Description
`source-name`	A regular expression that defines a set of source column names. This value is required.
`target-name`	A regular expression that defines a set of target column names. It can include references to groups captured by the `source-name` expression.
`exclude`	Indicates whether or not the matched columns should be excluded from the compare pair. The options are: `"true"` - The matched columns should be excluded. `"false"` - The matched columns should be included. This is the default.
`type`	Indicates the type of the column. The options are: `"key"` - The column is used as a key. `"hash"` - The column is compared using a hash value. This is the default value. `"literal"`- The column is a literal value.
`format`	Specifies a format to override the comparison format that would normally be used.
`scale`	Specifies a scale to override the default scale for the comparison.
`precision`	Specifies a precision to override the default precision used for the comparison.
`timezone`	Specifies a time zone to override the default time zone of the comparison.
`optional`	Indicates whether the column mapping is optional. For example, mapping will not fail if the base tables do not have the column patterns specified. Default is `"false"`.

colfilter

The `colfilter` element defines a set of columns to be included or excluded. It is used to specify the names of the columns to use as filtering criteria.

The following element describes the `colfilter` element:

Table 5-4 colfilter Element

Attribute	Description
`colfiltercol`	Specifies a set of columns to be included or excluded.

The following attribute describes the `colfilter` element:

Table 5-5 colfilter Attribute

Attribute	Description
`type`	Specifies whether to include the columns or exclude them. The options are `include` or `exclude`; the default is `include`. This is a required attribute.

Example

This example excludes `COL3` and `COL5` for the table `TABLE_NAME` from the generated compare pair.

```
<filter type="include" table="TABLE_NAME">
    <colfilter type ="exclude">
        <colfiltercol name="COL3" />
        <colfiltercol name="COL5" />
    </colfilter>
</filter>
```

colfiltercol

The `colfiltercol` element defines a set of columns to be included or excluded. It is used to specify the names of the columns to use as filtering criteria.

The following attribute describes the `colfiltercol` element:

Table 5-6 colfiltercol Attribute

Attribute	Description
`name`	A regular expression that defines a set of source column names. This is a required attribute.

compare-pair

The `compare-pair` element specifies a set of compare pair items. As in the Oracle GoldenGate Veridata Web User Interface, the compare pairs default to system mapped keys and columns.

The following elements can be nested within the `compare-pair` element:

Table 5-7 compare-pair Elements

Element	Description
`enscribe-info`	One or more sets of information used when comparing NonStop Enscribe files.
`sql-partition`	One or more specifications of a subset of rows within the table.
`enscribe-key`	One or more specifications of a subset of records within an Enscribe file.
`key-column`	A set of columns to be used as the user-defined key for the comparison.
`column`	One or more definitions of a set of columns to be included.
`excluded-column`	Defines a set of columns to excluded from the a compare pair when the compare pair uses system mapped columns.
`delta-config`	Defines the delta processing configuration for the compare pair. The maximum is to add it once per compare pair.

The following attributes describe the `compare-pair` element:

Table 5-8 compare-pair Attributes

Attribute	Element
`name`	An expression defining the name of the compare pair. This expression can include groups captured with `source-table` expressions and target table group `$0`.
`source-table`	A regular expression that defines the table or tables to be compared. See "Regular Expression Grouping" later in this section for more detail. The default is to match all tables.
`target-table`	A regular expression that defines the target tables for the comparison. This may contain references to groups captured by the source table expression. The default is $0 for the full source table name.
`source-schema`	The name of the default schema for the source tables referenced for the compare pair. The default is the value specified for the `group`. For SQL/MP, this is the subvolume of the SQL catalog. This is not used with Enscribe files.
`target-schema`	The name of the default schema for the target tables referenced for the compare pair. The default is the value specified for the `group`. For SQL/MP, this is the subvolume of the SQL catalog. This is not used with Enscribe files.
`source-catalog`	The default catalog for the source tables referenced in this compare pair. For SQL/MP, this is the volume of the SQL catalog. This is not used for Oracle, DB2, Enscribe, or Teradata.
`target-catalog`	The default catalog for the source tables referenced in this compare pair. For SQL/MP, this is the volume of the SQL catalog. This is not used for Oracle, DB2, Enscribe, or Teradata.

Table 5-8 (Cont.) compare-pair Attributes

Attribute	Element
exclude	Indicates whether or not the compare pair should be included in the group element. This can be used to remove a compare pair generated by an earlier compare pair element. The options are: "true" - Exclude the compare pair. "false" - Include the compare pair. This is the default.
source-file-pattern	The default file pattern for the source if the data source is Enscribe or SQL/MP.
target-file-pattern	The default file pattern for the target if the data target is Enscribe or SQL/MP.
source-pkey	The name of the unique index to use as the source portion of the user-specified primary key. The default is no user-specified index name.
target-pkey	The name of the unique index to use as the target portion of the user-specified primary key. The default is the value of the source-pkey.
delta-processing	Indicates whether or not delta processing is enabled for this compare pair. The options are: "true" - delta processing is enabled. "false" - delta processing is not enabled. This is the default.
profile-name	The name of the profile to use when running the compare-pair comparison.
system-key	If the compare pair has no column elements and no specified source-pkey, Oracle GoldenGate Veridata will select the most appropriate primary key or unique index to use. The options are: "true" - Oracle GoldenGate Veridata selects the key if it is not defined. This is the default. "false" - Oracle GoldenGate Veridata does not select the key.
system-columns	Indicates that the compare pair contains column elements with the type attribute set to key, so the generated compare pair will have user-defined columns for the key. The options are: "true" - Compare pair has key column elements. This is the default. "false" - Compare pair does not have key column elements.
wildcard	Specifies the pattern matching method that is used. The options are: "ggs" - Use the typical Oracle GoldenGate pattern that matches an asterisk (*) to any number of characters. "regex" - Use regular expressions for matching. "default" - Use the setting for the configuration. This is the default.

Regular Expression Grouping

Regular expression grouping can be used to capture the parts of the source table names to be used for matching the target table name. You can do this by changing the

`wildcard` attribute should be changed to `regex`. Groups to be matched are referenced as `$1`, `$2`, `$3` and so on. Group `$0` matches the entire source table name.

Examples of matching groups include:

- `P(.*)` - Matches table names that begin with `P`. It captures the variable portion in `$1`. This matches table `PROSPECTS`.
- `[^PV].*` - Matches table names that do *not* begin with `P` or `V`. This does not match the table `PROSPECTS`, but does match the table `REGIONS`.
- `([P-R])(.*)` - Matches table names starting with `P`, `Q`, or `R` and captures the initial letter in group `$1` and the rest of the name in group `$2`. Groups are defined by parenthesis pairs. Group numbers are defined by the count of left parenthesis. Group `$1` starts at the first left parenthesis and group `$2` starts at the second parenthesis.

Captured groups (`$n`) are then used in expressions for selecting the target tables.

Example

The following example describes the `key-only` compare-pair. It's source tables are defined in the `"test"` schema and target tables in the `"other"` schema. It creates a compare pair in which the source table name begins with `S` and target table name begins with `T`. For example, `S_TABLE` and `T_TABLE`, where `S_TABLE` is a table in schema "test" and `T_TABLE` is table in schema "other". It also excludes all non-key columns in the generated compare pairs.

```
<configuration>
    <connection name="source" host="somehost"
               .... use-ssl="true">
        <description>
             <![CDATA[
                Group SQL Scripting Source Connection
              ]]>
          </description>
     </connection>
...
...
</configuration>
```

connection

The `connection` element defines a connection to a source or target comparison database through an Oracle GoldenGate Veridata agent.

The following elements can be nested within the `connection` element:

Table 5-9 connection Elements

Element	Description
`description`	Provides a description of the connection.
`conn-properties`	Defines the connection properties for a connection.

The following attributes describe the `connection` element:

Table 5-10 connection Attributes

Attribute	Description
`name`	A name that identifies the connection. This is a required attribute.
`host`	The name of the system on which the Oracle GoldenGate Veridata agent is running.
`port`	The port number of the system on which the agent is running.
`user`	The user name the agent uses to connect to the database.
`password`	The password the agent uses to connect to the database.
`repairUser`	The database user with privileges to perform repair operations. See Database Privileges for the Agent Component.
`repairPassword`	The password for the repairUser.
`agent-timeout`	The amount of time Oracle GoldenGate Veridata will wait before timing out when sending requests to the agent.
`truncate-spaces`	Either `"true"` or `"false"` to indicate whether or not spaces will be removed from the end of character columns. The default is `"true"` to truncate spaces.
`fetch-size`	(Oracle only) The number of rows fetched in each batch.
`use-ssl`	Defines using SSL communication between the Veridata Agent and the Server. The default is `"true"`.

Example

The following example identifies the connection named `source`.

```
<configuration>
   <connection name="source" host="somehost"
    port="7850" user="somename" password="somepw"repairUser="veridata1"
repairPassword="veridata1" agent-timeout="4000" truncate-spaces="false" fetch-
size="3" use-ssl="true">
      <description>
             <![CDATA[
                Group SQL Scripting Source Connection
            ]]>
...
...
      </description>
   </connection>
.
.
</configuration>
```

conn-properties

The `conn-properties` element provides additional connection to a source or target comparison database elements.

The following attributes can be nested within the `conn-properties` element:

Table 5-11 conn-properties

Element	Description
`datatype-name`	Specifies the data type for which properties have changed.
`format`	Specifies the Veridata comparison format to be used for comparison.
`precision`	Specifies the precision to be applied to the comparison.
`scale`	Specifies the scale to be applied to the comparison.
`timezone`	Timezone name is same as in the Veridata GUI.

delta-config

The `delta-config` element defines the delta processing configuration for the specified compare pair. It can be used once per compare pair. This element can appear once or not at all depending on the type of configuration you want. When the source or target configuration specified, the corresponding column-name attribute and query element are mandatory.

The following elements describe the `delta-config`:

Table 5-12 delta-config Elements

Attribute	Description
`source-config`	Provides source side configuration for delta processing.
`target-config`	Provides target side configuration for delta processing.
`query`	Specifies the query for delta processing.

Example

This example creates a compare pair with delta processing enabled. Delta processing is enabled on `COL1` of `SYSMAPPING1` table for both source and target side. The SQL query is defined within the "query" tag.

```
<configuration validation="required">
   .
   .
   <group name="testGroup" source-conn="sourceConn" target-conn="targetConn" source-
schema="sourceSchema" target-schema="targetSchema">
        <compare-pair source-table="SYSMAPPING1" target-table="SYSMAPPING1"
name="sameTables" delta-processing="true" >
                  <delta-config>
                        <source-config column-name="COL1">
                              <query><![CDATA[ SELECT MAX(COL1) from
```

```
SYSMAPPING1 ]]></query>
                        </source-config>
                        <target-config column-name="COL1">
                              <query><![CDATA[ SELECT MAX(COL1) from
SYSMAPPING1 ]]></query>
                        </target-config>
                  </delta-config>
            </compare-pair>
    </group>
      .
      .
</configuration>
```

description

The `description` element is free-form text that can be used to attach a description to the containing element. It has no associated attributes.

Example

The following example provides a description for the connection named `source`.

```
<configuration>
   <connection name="source" host="somehost"
    port="7850" user="somename" password="somepw"
      <description>
         <![CDATA[
           This connection is used when the Veridata agent connects
           to the source.
         ]]>
      </description>
   </connection>
.
.
.
</configuration>
```

enscribe-info

The `enscribe-info` element provides additional information used to compare NonStop Enscribe records at the field level.

The following elements can be nested within the `enscribe-info` element:

Table 5-13 enscribe-info Elements

Element	Description
`expanddl`	Describes the rules that are used when applying the DDL.

The following attributes describe the `enscribe-info` element:

Table 5-14 enscribe-info Attributes

Attribute	Description
side	Indicates whether the information applies to the source or the target table. The options are: "source" to specify the source table. This is the *default*. "target" to specify the target table.
dictionary	The volume and subvolume containing the data dictionary.
record	The name of the record in the data dictionary.

enscribe-key

The enscribe-key element defines the key that is to be used for Enscribe files. The enscribe-key element defines a delta processing that can used in a where clause on the initial comparison query.

The following attributes describe the enscribe-key:

Table 5-15 enscribe-key Attributes

Attribute	Description
name	A name that identifies the key. This is a required attribute.
start-key	The key that is to be used to begin reading the Enscribe file. This is a required entry.
end-key	The key of the last Enscribe record that should be read. This is a required entry.
format	Specifies the format of the Enscribe key. The options are: "ascii" - The format of the key is ASCII. This is the default. "hexadecimal" - The format of the key is hexadecimal.
side	Indicates whether the partition should be applied at the source database, the target database, or both databases.
default	Indicates whether this is the default partition. This is equivalent to the "use at run time" indicator on the UI. The default is both.

Examples

```
<enscribe-key name = "Part1" end-key ="1000" format ="hexadecimal" default ="false"
side="source"/>
<enscribe-key name = "Part1" start-key ="001" format ="hexadecimal" default ="false"
side="target"/>
<enscribe-key name = "Both" start-key ="001" end-key ="1000" default ="true"/>
```

excluded-column

The `excluded-column` element defines a set of columns to be excluded from a compare pair when the compare pair uses system mapped columns.

The following attribute describes the `excluded-column` element:

Table 5-16 excluded-column Attributes

Attribute	Description
`name`	A regular expression that defines a set of source column names. This is a required attribute.

expanddl

The `expanddl` element describes the rules used when applying the DDL.

The following attributes describe the `expanddl` element:

Table 5-17 expanddl Attributes

Attribute	Description
`expandGroupArrays`	Whether or not to expand group arrays. The options are: `"true"` to expand the array. This is the default. `"false"` not to expand the array.
`redefined-columns`	Whether or not to include redefined columns. The options are: `"include"` - Includes redefined columns `"omit"` - Leaves out redefined columns. This is the default.
`resolvedups`	Specifies how to resolve duplicates that result when the array is expanded. The options are: `"appendIndex"` - Adds a unique numeric index to the end of the duplicate. This is the default. `"appendAlphaIndex"` - Adds an alpha character index to the end of the duplicate. `"prependGroup"` - Prefixes the name of the array group to the duplicate.
`ddl-separator`	The character separator for defining array output into columns. An example is the dash used in FIELDX-3, which is the third occurrence of `FIELDX` in the array. The options are: `"none"` - There is no separator. This is the *default*. `"dash"` - Use a dash (-) as the separator. `"bracket"` - Use brackets [] as the separator. `"underscore"` - Use underscore (_) as the separator. `"double-underscore"` - Use double underscore (__) as the separator.
`zero-fill-length`	Prepends zeros to adjust the number of the occurrence. The value is the number of digits enclosed in quotation marks. "0" is the default.

Table 5-17 (Cont.) expandddl Attributes

Attribute	Description
`fix-long-names`	Whether to fix the names that result from resolving duplicates if they exceed the `max-col-name-length`. The options are: `"true"` - Fix the names that exceed the maximum. This is the default. `"false"` - Do not change the names that exceed the maximum.
`max-col-name-length`	The maximum length allowed for a column name. The entry is a number within quotation marks. The default is `"120"`.

filter

The `filter` element defines a set of schemas and tables to either be included or excluded.

When using include filters, at least one filter must be matched before a table can be included in a compare pair. When a table matches a include filter, the include filter's `colfilter` is used to specify the columns for the generated compare pair.

When using exclude filters, a table is excluded if it matches any exclude filter. Include filters can include a `colfilter` element, which contains a list of columns to include or exclude.

Instead of schema and table filters, NonStop platforms use file pattern filters. The file pattern is any valid NonStop platform file name pattern.

The schema and table name can use wildcards.

The following attribute describes the `filter` element:

Table 5-18 filter Attributes

Attribute	Description
`type`	Specifies either to include or exclude schemas and tables. Valid values are `include` or `exclude`.
`catalog`	Specifies the default catalog name.
`schema`	Specifies the schema name.
`table`	Specifies the table name.
`file-pattern`	For NonStop platforms only, specifies the file patter filter.

Example

When the source and target schemas have `CHAR_TYPES3`, `INT_TYPE1`, and `INT_TYPE2` tables, then the following filters only create compare pairs for tables `CHAR_TYPES1` and `CHAR_TYPES3`. The `CHAR_TYPES2` table is excluded because of exclude filter and `INT_TYPE1` and `INT_TYPE2` are excluded because they were not part of include filter.

```
<group
      ..
        <filter type="include" table="CHAR_TYPES*" />
```

```
        <filter type="exclude" table="CHAR_TYPES2" />
        <compare-pair source-table="*" target-table="*">
        </compare-pair>
      ..
</group>
```

group

The `group` element defines a set of compare pairs that all have the same source and target database connections. These compare pairs also have other properties in common.

The following elements can be nested within the `group` element.:

Table 5-19 Group Elements

Element	Description
`description`	Provides a description of the group.
`filter`	One or more filter specifications, which allows table name filtering at the group level.
`sql-partition`	One or more specifications of a subset of rows within the table.
`enscribe-key`	One or more specifications of a subset of records within an Enscribe file.
`compare-pair`	Defines one or more compare pairs. The `compare-pair` elements are added to the group in the order they are specified. If the same compare pair fits the criteria of another specification in the group, the first compare pair will be used.

The following attributes describe the `group` element:

Table 5-20 Group Attributes

Attribute	Description
`name`	A name that identifies the group. This value is required.
`source-conn`	The name of the connection to the source database. This can reference an existing connection in the Oracle GoldenGate Veridata repository or a connection previously defined in this configuration. This attribute is required if it references an existing connection in the repository.
`target-conn`	The name of the connection to the target database. This can reference an existing connection in the Oracle GoldenGate Veridata repository or a connection previously defined in this configuration. This attribute is required if it references an existing connection in the repository.
`source-schema`	The name of the default schema for the source tables referenced in the compare pairs that make up the group.

Table 5-20 (Cont.) Group Attributes

Attribute	Description
`target-schema`	The name of the default schema for the target tables referenced in the compare pairs that make up the group.
`source-catalog`	The default catalog for the source tables referenced in this group.
`target-catalog`	The default catalog for the target tables referenced in this group.
`validation`	Specifies the type of validation that will be used for the configurations. The options are: "`required`" - All compare pairs must be successfully validated before any pairs are added to the repository. "`omit-failures`" - Successfully validated compare pairs are added to the repository and compare pairs that cannot be validated are ignored. "`none`" - Compare pairs are added to the repository without any validation. If this option is selected the Oracle GoldenGate Veridata Web User Interface should be used to review and fix validation problems. "`default`" - Use the type of validation specified for a higher level, such as the `configuration` element. This is the default.
`source-file-pattern`	The default file pattern for the source if the data source is Enscribe or SQL/MP.
`target-file-pattern`	The default file pattern for the target if the data target is Enscribe or SQL/MP.

Example

```
<group name="weekly-tables" source-conn="source" target-conn"="target">
   <description>
   .
    .
    .
   </description>
   <sql-partition>
    .
    .
   </sql-partition>
   <compare-pair>
    .
    .
    .
</compare-pair>
</group>
```

job

The `job` element defines an Oracle GoldenGate Veridata comparison job.

The following elements can be nested within the `job` element:

Table 5-21 job Elements

Element	Description
`description`	Provides a description of the job.
`group`	The name of the group associated with the job. This can be a new group or a previously defined group.

The following attributes describe the `job` element:

Table 5-22 job Attributes

Attribute	Description
`name`	A name that identifies the job. This is a *required* attribute.
`source-conn`	The name of the connection to the source database. This can reference an existing connection in the Oracle GoldenGate Veridata repository or a connection previously defined in this configuration. This attribute is required if it references an existing connection in the repository. The job `source-conn` is used to override the source connection specified for the groups included in the job.
`target-conn`	The name of the connection to the target database. This can reference an existing connection in the Oracle GoldenGate Veridata repository or a connection previously defined in this configuration. This attribute is used to override the target connection for the groups included in the job.
`profile`	The default profile to use when running the job.

Example

```
<job name="all-groups" profile="server-sort">
    <group name="all-tables"/>
    <group name="selected-tables"/>
</job>
```

profile

The `profile` element defines the connection properties of a comparison job connection.

The following elements can be nested within the `profile` element:

Table 5-23 profile Elements

Element	Description
`description`	Provides a description of the profile.

Table 5-23 (Cont.) profile Elements

Element	Description
`profile-general`	Defines the profile parameters that control the output options.
`sorting-method`	Defines the profile parameters that control the sorting method and memory management. The data is sorted to match keys (or a key specification) so that the correct source and target rows are compared.
`initial-compare`	Defines the profile parameters that control the parameters for the job that performs the initial compare step
`confirm-out-of-sync`	Specifies the profile parameters that control the parameters for the job that performs the confirmation step
`repair`	Specifies the profile parameters that control the parameters for the repair job.

The following attributes describe the `profile` element:

Table 5-24 profile Attributes

Attribute	Description
`name`	A name that identifies the profile. This is a required attribute.

Example

This example creates profile named "userDefinedProfile". The parameter names like "oos-format", "sort-method" are described in the table (link for table is in another pin)

```
<configuration validation="required">
   .
   .
<profile name="userDefinedProfile">
    <profile-general>
        <param name="oos-format" value="xml" />
        <param name="oos-xml-chunk-size" value="1000" />
    </profile-general>
    <sorting-method>
        <param name="sort-method" value="server" />
    </sorting-method>
</profile>
      .
      .
</configuration>
```

key-column

The `key-column` element defines a set of columns to be used as the user defined key for the comparison job.

The following attributes describe the `key-column` element:

Table 5-25 profile Attributes

Attribute	Description
source-name	A regular expression that defines a set of source column names. This value is required.
target-name	A regular expression that defines a set of target column names. It can include references to groups captured by the source-name expression.
format	Specifies a format to override the comparison format that would normally be used.
scale	Specifies a scale to override the default scale for the comparison.
precision	Specifies a precision to override the default precision used for the comparison.
timezone	Specifies a time zone to override the default time zone of the comparison.

profile-general

The profile-general element provides parameters to control the output options.

The data is sorted to match keys (or a key specification) so that the correct source and target rows are compared.

The following elements can be nested within the profile-general element:

Table 5-26 profile-general Element

Element	Description
param	Defines the parameter to change for the profile.

sorting-method

The sorting-method element provides parameters for sorting method and memory management. The data is sorted to match keys (or a key specification) so that the correct source and target rows are compared.

The following elements can be nested within the sorting-method element:

Table 5-27 sorting-method Element

Element	Description
param	Defines the parameter to change for the profile.

initial-compare

The `initial-compare` element provides parameters for the process that performs the initial compare step.

The following elements can be nested within the `initial-compare` element:

Table 5-28 initial-compare Element

Element	Description
`param`	Defines the parameter to change for the profile.

confirm-out-of-sync

The `confirm-out-of-sync` element provides parameters for the process that performs the confirmation step.

The following elements can be nested within the `confirm-out-of-sync` element:

Table 5-29 confirm-out-of-sync Element

Element	Description
`param`	Defines the parameter to change for the profile.

param

The `param` element defines the parameters that are used for configuring profile options.

The following attributes describe the `repair` element:

Table 5-30 param Attributes

Attribute	Description
`name`	The name of the parameter. This is a required attribute.
`value`	The value of the parameter

repair

The `repair` element provides parameters for the repair process.

The following elements can be nested within the `repair` element:

Table 5-31 repair Element

Element	Description
`param`	Defines the parameters that are used to configure the profile options.

sql-partition

The `sql-partition` element defines a boolean SQL expression that can be used in a `where` clause in the initial comparison query.

The following attributes describe the `sql-partition` element:

Table 5-32 sql-partition Attributes

Attribute	Description
`name`	A name that identifies the partition. This is a required attribute.
`side`	Indicates whether the partition should be applied at the source database, the target database, or both databases. The default is `"both"`.
`default`	Indicates whether this is the default partition. This is equivalent to the "use at run time" indicator on the UI. The default is `"false"`.

Example

```
<sql-partition name="replicate" default="true" side="source">
  <![CDATA[ replicated='false']]>
</sql-partition>
<sql-partition name="replicate" default="true" side="source">
  <![CDATA[ replicated='true']]>
</sql-partition>
```

6

Running Veridata GoldenGate Parameter Processing

Veridata GoldenGate Parameter Processing (VGPP) is a command line tool to help you manage your Extract or Replicat parameter files. This includes parameter and map and table statement handling.
This chapter includes the following sections:

Overview of the Command-Line Interface

An Oracle GoldenGate Veridata parameter file contains all of the information required to extract or apply replicated data. The data-propagated replication is controlled by settings in the Extract and Replicat parameter files. Typically, the Extract parameter file specifies the tables to be replicated, the Replicat parameter file controls column mapping and restricts the tables. The Veridata GoldenGate Parameter Processing command-line utility accepts one or two parameter files as input. One of the files *must* be a Replicat parameter file, and the other optional file must be an Extract parameter file.

Replication captures information about the transaction responsible for changing the data as well as the actual changed data. However, Oracle GoldenGate Veridata can detect only the current state of the source and target databases, so it cannot support parameters for transactional changes, such as the `INSERTDELETES` parameter. Veridata GoldenGate Parameter Processing uses the MAP statements from a REPLICAT file to generate compare pairs. Other information in the parameter file is used to determine the relevant MAP statements. Optionally, you can use the TABLE statements from the EXTRACT parameter to restrict the compare pairs that are generated.

Because Oracle GoldenGate Veridata supports only single-column mapping, the column mapping assumes one-to-one mapping between source columns and target columns.

You can:

- Reuse Replicat and Extract configurations in Oracle GoldenGate Veridata
- Avoid creating separate Replicat and Extract parameter files.

Running the Veridata GoldenGate Parameter Processing

You can run the Veridata GoldenGate Parameter Processing program if you have the correct operating system permissions.

1. Ensure that the parameter files you want to use are on the system where the Oracle GoldenGate Veridata is installed. If you need to copy the files from another system, these files should be copied as binary files so that the `CHARSET` parameter remains valid.
2. Run the command shell of the operating system.
3. Navigate to the *`VERIDATA_DOMAIN_HOME`*`/veridata/bin` directory.
4. Run the Veridata GoldenGate Parameter Processing.

Syntax

```
veridata_param_process{.bat|.sh} required_parameter [optional_parameter]
```

Required Parameters

The following are required; otherwise an error is returned.

```
[-noscripting |
-create |
-replace|
-update ]
[-wlport port ] |
-wluser user_name |
[-p <propfile>]
[-o <outputFile>]
[replicat_param_filename]
```

The `-wluser` option specifies the Oracle GoldenGate Veridata Server (server) user name to connect to the server. This server user should have the `veridataCommandLineUser` privilege to access and execute command-line operations. The user should also have the `veridataAdministrator` or `veridataPowerUser` privilege to successfully run jobs and to use the import and export utilities.

Optional Parameter

This is the optional parameter:

```
[extract_param_filename]
```

If you provide the Extract and Replicat file names, then give the Extract file name first, followed by the Replicat file name.

Table 6-1 VGPP Runtime Arguments

Argument	Description
`{ -noscripting | -create | -replace | -update}`	Specifies that `veridata_scripting` is not run with the generated configuration file. The `-create`, `-replace`, and `-update` options indicate that `veridata_scripting` should be run with the generated configuration file. In either case, the generated scripting configuration file can be used as input to the `veridata_scripting`. The default option is `-create`.
`-wluser`	Specifies the Oracle WebLogic user name that authenticates and connects to the server.
`-wlport`	Specifies the Oracle WebLogic Server port number. The default listening port is 8830.

Table 6-1 (Cont.) VGPP Runtime Arguments

Argument	Description
`-p`	Specifies a properties file containing additional information required for the configuration.
`-o`	Specifies the output file containing the generated scripting file. The default is *`replicat_name`*`_scripting.xml`; *`replicat_name`* is the value of the `REPLCAT` parameter. The optional *`extract_param_filename`* parameter specifies an `EXTRACT` parameter file containing source information for the comparison. The *`replicat_param_filename`* parameter is the `REPLICAT` parameter containing the target information.

Using a Property File

When the VGPP program is run, an optional property file can be specified. This file contains information that is not available in the Oracle GoldenGate parameter file and is required to generate a valid Veridata comparison configuration. The following are some of properties (information) that you can specify.

Table 6-2 Optional Parameters

Property Name	Comments
`source.connection.name`	The name of the Veridata agent/manager connection. This may be the name of an existing Veridata connection. The default is the Extract name. This is the only source connection property needed to reference an existing connection.
`source.connection.port`	The port for the source agent. This is required when the connection does not already exist.
`source.connection.host`	The host name where the source agent is running. This is required when the connection does not exist.
`source.connection.user`	This defaults to the user information in the extract parameter file. This is required when the connection does not exist.
`source.connection.password`	This defaults to the user information in the extract parameter file. This is required when the connection does not already exist. If property name is specified without a value, the scripting utility will prompt for the value when the scripting configuration is loaded into Veridata.
`source.catalog`	This is valid for Sybase, SQL Server, and Oracle consolidated databases. For Sybase and SQL Server, it is the database containing the source tables. For Oracle, it specifies the Oracle PDB to use when processing an Extract parameter. Statements not associated with this PDB are ignored. The default value is the first PDB reference in the file. The reference can be a `SOURCECATALOG` parameter or the first part of a three-part name in a `TABLE` statement.

Table 6-2 (Cont.) Optional Parameters

Property Name	Comments
extract.useansiquotes	Indicates whether or not the Extract parameter file follows the ANSI quotation specification. This is a Boolean value. The default value is true. This is the same as the GoldenGate core GLOBALS parameters USEANSISQLQUOTES | NOUSEANSISQLQUOTES.
extract.charset	The character set for the extract parameter file. This overrides any charset specified in the extract parameter file.
extract.trail	The trail file name to use when more than one trail file is specified in an extract parameter file. The default is the first trail file specified in the extract parameter file. When an extract parameter file contains multiple rmttail entries with the same name, Veridata will use the first occurrence.
target.connection.name	The name of the Veridata agent/manager connection. This may be the name of an existing Veridata connection. The default is the Replicat name. This is the only target connection property required to reference an existing connection.
target.connection.port	The port for the target agent. This is required when the connection does not already exist.
target.connection.host	The host name where the target agent is running. This is required when the connection does not exist.
target.connection.user	This defaults to the user information in the parameter file
target.connection.password	This defaults to the user information in the extract parameter file. If property name is specified without a value, the scripting utility prompts for the value when the scripting configuration is loaded into Veridata.
target.catalog	This is valid for Sybase and SQL Server. It is the database containing the target tables.
replicat.useansiquotes	Indicates whether or not the replicat parameter file follows the ANSI quotation specification. This is Boolean value. The default value is true. This is the same as the GoldenGate core GLOBALS parameters USEANSISQLQUOTES | NOUSEANSISQLQUOTES.
replicat.charset	The character set for the replicat parameter file. This overrides any CHARSET specification in the replicat parameter file.

Parameter Handling

This section describes the handling of all of the parameters allowed in an Oracle GoldenGate Extract or Replicat parameter file. Each keyword is either supported, unsupported, or ignored. A supported parameter is used to generate the Veridata configuration. An unsupported parameter is something that interferes with the Veridata configuration generation. When an unsupported parameter (INSERTDELETES) is specified, subsequent MAP parameters are ignored. An ignored parameter specifies a feature that is not applicable to Veridata configuration generation.

The following table contains the known parameters and the expected handling: if a parameter is not listed, it is ignored.

Table 6-3 Parameter Handling

GoldenGate Parameters	Veridata Support
CATALOGEXCLUDE	This parameter is ignored. Veridata only processes items from a single catalog.
CHARMAP	Unsupported.
CHARSET	Supported. This parameter is supported in parameter files and include/obey files. Veridata does not process GLOBALS files.
COLMATCH	Supported.
COMMENT | --	Supported.
DICTIONARY	Supported for NSK extract and replicat.
EXPANDDDL	Supported for NSK extract and replicat.
EXCLUDEWILDCARDOBJECTSONLY	Supported.
EXTRACT	Supported.
EXTTRAIL	Supported.
FILE | TABLE	Supported for NSK extract.
INCLUDE	Supported. When the file is not found by the specified path, VGPP will look for the file name in the same directory as the parameter file.
INSERTALLRECORDS	Unsupported. Ignore all subsequent MAP statements.
INSERTDELETES | NOINSERTDELETES	Unsupported | Supported. Ignore all MAP statements following an INSERTDELETES command until a NOINSERTDELETES command is found.
INSERTUPDATES | NOINSERTUPDATES	Unsupported| Supported: Ignore all MAP statements between the INSERTUPDATES and the NOINSERTUPDATES.
MACRO	Supported.
MACROCHAR	Supported.
MAP	Supported.
MAPEXCLUDE	Supported.
OBEY	Supported. The same as INCLUDE.
REPLICAT	Supported.
RMTTRAIL	Supported.
SCHEMAEXCLUDE	Supported.
SOURCECATALOG	Supported for Oracle consolidated databases.
TABLE | MAP	Supported. The details are explained in section 3.4.4
TABLEEXCLUDE	Supported.
UPDATEDELETES | NOUPDATEDELETES	Unsupported | Supported.
UPDATEINSERTS | NOUPDATEINSERTS	Unsupported | Supported.
USEANSISQLQUOTES | NOUSEANSISQLQUOTES	Supported.

Map and Table Statement Handling

Veridata will generate a compare pair element in the scripting configuration file for each Map statement in the Replicat parameter file. The generated scripting file will list the specific table mappings first, followed by the wildcard mappings, and finally the excluded mappings. This matches the behavior of the Oracle GoldenGate Replicat where specific mappings take precedence over wildcard mappings.

When the same source and target table specification appears in multiple `MAP` statements, the first occurrence will be used for the compare pair specification. The multiple occurrences can occur when the `MAP` statements use thread specifications and range filters.

The following table lists all of the keywords for the `MAP` and `TABLE` statements and support level in VGPP. `MAP` statements containing unsupported keywords will not generate a Veridata comparison configuration. Items marked with maybe indicate that more information is needed in order to determine the value for Veridata.

Table 6-4 Map and Table Statement Handling

Keyword	Veridata Support
`TARGET`	Supported.
`COLMAP`	Supported. Only simple source column to target column mapping is supported. Target columns mapped to functions or literals is excluded from the comparison configuration. The `USEDEFAULTS` keyword is supported. The `BINARYINPUT` keyword is ignored.
`COLS`	Supported.Results in an explicit column list in the generated compare pair configuration.
`COLSEXCEPT`	Supported. If an explicit column mapping does not exist, this results in system mapped columns with a list of omitted columns.
`COMPARECOLS`	Ignored
`COORDINATED`	Ignored.
`DICTIONARY`	Supported for NSK.
`TARGETDICT`	Supported for NSK.
`DEF`	Supported for NSK.
`TARGETDEF`	Supported for NSK.
`EVENTACTIONS`	Ignored.
`EXCEPTIONSONLY`	Unsupported.
`EXITPARAM`	Ignored.
`FETCHBEFOREFILTER`	Ignored
`FETCHCOLS|FETCHCOLSEXCEPT`	Ignored
`FETCHMODCOLS|` `FETCHMODCOLSEXCEPT`	Ignored
`FILTER`	Ignored.

Table 6-4 (Cont.) Map and Table Statement Handling

Keyword	Veridata Support
GETBEFORECOLS	Ignored.
HANDLECOLLISIONS | NOHANDLECOLLISIONS	Ignored
INSERTALLRECORDS	Unsupported.
INSERTAPPEND | NOINSERTAPPEND	Ignored
KEYCOLS	Supported.
MAPEXCEPTION	Ignored.
REPERROR	Ignored
RESOLVECONFLICT	Ignored.
SQLEXEC	Ignored
SQLPREDICATE	Ignored
THREAD	Ignored.
THREADRANGE	Ignored.
TOKENS	Ignored.
TRIMSPACES | NOTRIMSPACES	Supported.
TRIMVARSPACES | NOTRIMVARSPACES	Supported.
WHERE	Ignored.

VGPP Example

Below is an example of running the VGPP utility.The example shows usage of VGPP utility using a simple extract file, replicat file and property file.

```
./veridata_param_process.sh -noscripting -wluser veridata -wlport 8830 /scratch/ggcore/dirprm/extract.prm /scratch/ggcore/dirprm/replicat.prm -p properties.txt -o output.xml
```

Here:`extract.prm`- the extract param file. This is optional.

`replicat.prm`- the replicat param file. This is required.

If both the extract and replicat files are given, then the extract file should be given before replicat file.

`properties.txt`- this file contains properties related to veridata agents.

`output.xml`- this is the generated xml file which can be used with Veridata Scripting Import tool.

Sample Content

extract.prm

```
CHARSET utf-8
EXTRACT sqlParamSrcConnection
```

```
LOGALLSUPCOLS
RMTHOST localhost, MGRPORT 7000
RMTTRAIL sqlParamSrcConnection_trail
TABLE SOURCE.CHAR_*;
```

replicat.prm

```
CHARSET US-ASCII
REPLICAT sqlParamTrgConnection
ASSUMETARGETDEFS
MAP SOURCE.CHAR_TYPES, TARGET TARGET.CHAR_TYPES_NOTNULL, COLMAP ( USEDEFAULTS,
NCHAR_COL = NVARCHAR_COL, NVARCHAR_COL = NCHAR_COL, KEY_COL = CHAR_COL, CHAR_COL =
KEY_COL ),KEYCOLS (KEY_COL, CHAR_COL);
```

properties.txt

```
source.connection.host=localhost
source.connection.name=sqlParamSrcConnection
source.connection.port=7860
source.connection.user=source
source.connection.password=source


target.connection.host=localhost
target.connection.name=sqlParamTrgConnection
target.connection.port=7861
target.connection.user=target
target.connection.password=target
```

Generated output.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration SYSTEM "configuration.dtd">
<!--
Hostname : localhost
OGGV-30003: Extract filename : /scratch/ggcore/dirprm/extract.prm
OGGV-30002: Replicat filename : /scratch/ggcore/dirprm/replicat.prm
January 3, 2017 4:33:40 AM PST
-->
<configuration wildcard="ggs" operation="create" validation="required">
 <connection name="sqlParamSrcConnection" host="localhost" port="7860" user="source"
password="source"></connection>
 <connection name="sqlParamTrgConnection" host="localhost" port="7861" user="target"
password="target"></connection>
 <group name="sqlParamSrcConnection_sqlParamTrgConnection" source-
conn="sqlParamSrcConnection" target-conn="sqlParamTrgConnection">
  <filter type="include" schema="SOURCE" table="CHAR_*"></filter>
  <compare-pair source-schema="SOURCE" source-table="CHAR_TYPES" target-
schema="TARGET" target-table="CHAR_TYPES_NOTNULL">
  <key-column target-name="KEY_COL" source-name="CHAR_COL"></key-column>
  <key-column target-name="CHAR_COL" source-name="KEY_COL"></key-column>
  <column source-name="*" type="hash"></column>
  <column target-name="NCHAR_COL" source-name="NVARCHAR_COL" type="hash"></
column>
  <column target-name="NVARCHAR_COL" source-name="NCHAR_COL" type="hash"></column>
 </compare-pair>
</group>
<job name="sqlParamSrcConnection_sqlParamTrgConnection">
 <group name="sqlParamSrcConnection_sqlParamTrgConnection"></group>
```

```
 </job>
</configuration>
```

7

Oracle GoldenGate Veridata Server Configuration Parameters

Learn how to use parameters to adjust different aspects of the sort memory configuration when using server-side sorting.
This chapter includes the following sections:

Overview of the Server Memory

Oracle GoldenGate Veridata Server uses virtual memory in the following ways:

- **Server memory for basic operation**. This is the amount of virtual memory that the Veridata server and web components need to operate. It stores object pools, database access libraries, and other information. This is usually about 200 MB.
- **Sort memory**. This is the memory that is used when server-side sorting is used. The virtual memory for sorting is allocated for the entire comparison, not per thread. The rows are read from the agent and submitted to be sorted. The sorting occurs in a thread that is separate from the thread that reads from the agent, and the sort may use more threads to work in parallel. Once all the rows from the agent are submitted to the sort process, the server process retrieves the sorted rows from the sort for comparison.
- **Row hash queue memory**. This is the memory that buffers data between the agent processes, the sort process, and the server process. A comparison that uses database sorting requires a single queue each for the source and target. Each queue has a capacity of 20 MG. The memory usage by the queues is affected by the relative speed of the comparison and by the data coming from the agent. The relative speed between the two agents also affects the memory usage. A larger differential in speed increases the amount of memory that is used, because the queue needs to buffer the data.
- **MOOS queue memory.** This is the memory that holds potentially out-of-sync records between the initial comparison and confirmation steps of a comparison. The size of the MOOS queue is limited to 50K of records. Memory usage is also dependent on the width of each record.
- **IPC buffer memory**. This is the memory that is used to exchange messages between the server and the agent.
- **Scratch runtime transient memory**. This is virtual memory space.

The amount of memory that can be used by the sort process cannot be greater than the minimum of:

- System physical memory
- Available memory in swap
- Java boot option `-Xmx` maximum memory setting

Estimating Memory Usage

The maximum amount of memory available to Oracle GoldenGate Veridata is specified by the Java boot option `-Xmx`. When server-side sorting is used, a large portion of this memory is reserved for sorting during comparisons. This reserved amount is controlled by the `server.max_sort_memory` configuration parameter.

When a comparison is run, two buffers are allocated from the reserved sort memory. Each of these is equal to the size specified as **Maximum Memory Usage (MB)**. To access this setting click the **Edit** option from the Profile Configuration screen, then **Sorting Method** from the Profile settings categories.

To Estimate Memory based on the Number of Concurrent Comparisons

The maximum amount of memory that can be used for any comparison is set by the parameter `server.max_sort_memory`. The `-Xmx` Java boot option should be set large enough to allow the desired number of concurrent comparisons.

The maximum number of concurrent comparisons is defined by the `server.max.concurrent_comparison_threads` configuration parameter. Therefore the maximum amount of sort memory can be as large as:

`server.max_sort_memory` * `server.max_concurrent_comparison_threads`

For example, if you set `server.max_concurrent_comparison_threads` to allow 10 concurrent comparisons and leave `server.max_sort_memory` set to the default value of 100 MB, you will need 1 GB of available memory.

To Estimate the Amount of Memory Used per Row

Refer the section "Disk and Memory Requirements for the Server Component" in *Installing and Configuring Oracle GoldenGate Veridata* for the calculation to estimate the amount of memory used per row.

How to Set a Parameter

To set a parameter, edit its entry in the `veridata.cfg` file. This file is stored in the `DOMAIN_HOME/config/veridata` directory within the Oracle GoldenGate Veridata Server installation directory.

Open an Oracle service request before changing these parameters. For more information, go to `http://support.oracle.com.`

Parameter Descriptions

This section describes the parameters that can be set in the `veridata.cfg` file. These parameters are grouped under the following categories:

- Server Parameters
- Parameters for Configuring SSL Communication

- Parameters for Veridata Command-Line Utility
- Parameters for Report File Encryption

Server Parameters

This section defines the following configurable parameters for your Veridata Server:

- server.veridata_data
- server.persistence_db_type
- server.meta_session_handle_timeout
- server.max_concurrent_jobs
- server.max_concurrent_comparison_threads
- server.max_sort_memory
- server.concurrent.writers
- server.concurrent.readers
- server.number_sort_threads

server.veridata_data

The directory that contains Oracle GoldenGate Veridata reports.

Syntax

```
server.veridata_data path
```

where *path* is a relative or absolute path for the directory where Veridata reports will be stored.

> **Note:**
>
> If you specify a relative path for the data directory, you need not start the path with a forward (/) or backward (\) slash. The path will be relative to the Veridata domain home directory.

Default Value

```
veridata/reports
```

That means the default data directory is *VERIDATA_DOMAIN_HOME*`/veridata/reports`.

server.persistence_db_type

This parameter defines the persistence database type.

Syntax

```
server.veridata_data database_type
```

where *database_type* is the persistence database type. The options are:

- ORACLE_OCI
- MS_SQL

Default Value

ORACLE_OCI

server.meta_session_handle_timeout

This parameter defines the meta-session handle timeout in seconds.

Syntax

```
server.meta_session_handle_timeout seconds
```

Example

```
server.meta_session_handle_timeout 600
```

Default Value

900

server.max_concurrent_jobs

This parameter specifies the maximum number of jobs that can be run simultaneously.

Syntax

```
server.max_concurrent_jobs number_of_jobs
```

Example

```
server.max_concurrent_jobs 200
```

Default Value

100

server.max_concurrent_comparison_threads

Sets the maximum number of concurrent comparisons that can be executed. In general, the amount configured by the server is the optimal value, given the machines resources. You can lower this number to reduce the impact of the server on your system. When this limit is reached, no new comparisons will start until an active comparison completes.

Syntax

```
server.max_concurrent_comparison_threads {default | number}
```

- `default` allows Oracle GoldenGate Veridata to compute the maximum number of concurrent threads and available resources. The default value is the server.max_sort_memory.

- *number* is a positive integer that sets the maximum number of concurrent comparison threads.

Example

```
server.max_concurrent_comparison_threads 100
```

Default Value

The default value is the maximum of four or the number of available CPUs.

server.mapped_sort_buffers

Indicates whether sort buffers are allocated as a memory mapped file or allocated on the JVM heap.

Syntax

```
server.mapped_sort_buffers [true|false]
```

Example

```
server.mapped_sort_buffers true
```

Default Value

The default is `true`. If an error occurs during initialization, Oracle GoldenGate Veridata uses the JVM heap.

server.max_sort_memory

Sets the maximum amount of sort virtual memory that is available to all running comparisons that use server-side sorting. When a JVM heap sort is allocated using the Java boot option `-Xmx` maximum memory, the default setting is the available heap size less the 200 MB needed for basic tasks. When memory mapped file sort is used, the default is 2G. You can limit this amount to make more memory available for the Oracle GoldenGate Web User Interface.

If a comparison does get enough virtual memory, the currently available sort virtual memory gets decremented by the amount that the comparison reserves. When a comparison completes, it increments the amount of available sort virtual memory by the amount of sort virtual memory that it had reserved.

Syntax

```
server.max_sort_memory {default | number{M | m}}
```

- `default` allows Oracle GoldenGate Veridata to define a maximum value that is dependent on the operating system.
- *number*`{M | m}` specifies a value in megabytes. For example, 1000M means a limit of 1000 megabytes. If this number exceeds the amount of available memory, the value will be reduced to the amount of available memory.

Example

```
server.max_sort_memory 1000M
```

Default Value

The system calculates the default size based on the available virtual memory.

server.concurrent.writers

This parameter specifies the number of writer threads per sort directory.

Syntax

```
server.concurent.writers number
```

Example

```
server.concurent.writers number
```

Default Value

The maximum of 4 or one quarter of the number of available CPUs.

server.concurrent.readers

This parameter specifies the number of reader threads for the entire server.

Syntax

```
server.concurent.readers number
```

Example

```
server.concurent.readers number
```

Default Value

The maximum of 4 or one quarter of the number of available CPUs.

server.number_sort_threads

This parameter specifies the number of threads used to sort input buffers from the Veridata Agent.

> **Note:**
>
> The value of `server.number_sort_threads` should not be greater than the number of available processes.

Syntax

```
server.number_sort_threads number
```

Example

```
server.number_sort_threads number
```

Default Value

The maximum of 4 or one quarter of the number of available CPUs.

Parameters for Configuring SSL Communication

This section defines the parameters that you can use to configure SSL communication between your Veridata Server and Veridata Agents:

- server.useSsl
- server.ssl.client.allowTrustedExpiredCertificates
- server.ssl.client.identitystore.keyfactory.alg.name
- server.ssl.client.truststore.keyfactory.alg.name
- server.ssl.algorithm.name

server.useSsl

This parameter specifies whether SSL is enabled for communication between the Veridata Server and all Veridata Agents.

Syntax

```
server.useSsl [true|false]
```

Example

```
server.useSsl true
```

Default Value

The default value is false.

server.ssl.client.allowTrustedExpiredCertificates

If the value of this parameter is set to true, Veridata Server allows SSL communication between the agent and the server when a trusted certificate expires.

> **Note:**
>
> The parameter is not applicable if you are running IBM's JVM.

Syntax

```
server.ssl.client.allowTrustedExpiredCertificates [true|false]
```

Example

```
server.ssl.client.allowTrustedExpiredCertificates false
```

Default Value

The default value is true.

server.ssl.client.identitystore.keyfactory.alg.name

This parameter specifies a name for the identity store key factory algorithm used for SSL communication.

Syntax

```
server.ssl.client.identitystore.keyfactory.alg.name=algorithm_name
```

Example

```
server.ssl.client.identitystore.keyfactory.alg.name=IbmX509
```

If you are running on IBM's JVM, set the value to IbmX509.

Default Value

The default value is SunX509.

server.ssl.client.truststore.keyfactory.alg.name

This parameter specifies a name for the trust store key factory algorithm used for SSL communication.

Syntax

```
server.ssl.client.truststore.keyfactory.alg.name=algorithm_name
```

Example

```
server.ssl.client.truststore.keyfactory.alg.name=IbmX509
```

If you are running on IBM's JVM, set the value to IbmX509.

Default Value

The default value is SunX509.

server.ssl.algorithm.name

This parameter specifies algorithm used for SSL communication.

Syntax

```
server.ssl.algorithm.name=algorithm_name
```

Example

```
server.ssl.algorithm.name=TLS
```

Default Value

The default value is TLS.

Parameters for Veridata Command-Line Utility

This section defines the following configurable parameters for your Veridata Server:

- veridata.cli.run_from_managed_server
- veridata.cli.managed_server_name
- veridata.cli.server.listenAddress
- veridata.cli.server.timeout.seconds

veridata.cli.run_from_managed_server

To run the Veridata command-line utility from the Veridata Managed Server, set this parameter value to `true`.

Syntax

```
veridata.cli.run_from_managed_server [true|false]
```

Example

```
veridata.cli.run_from_managed_server false
```

Default Value

The default value is true.

veridata.cli.managed_server_name

This parameter specifies the name of the Veridata Managed Server.

Syntax

```
veridata.cli.managed_server_name server
```

Example

```
veridata.cli.managed_server_name VERIDATA_server2
```

Default Value

The default name of the managed server is `VERIDATA_server1`.

veridata.cli.server.listenAddress

This parameter specifies the listening address of the host machine for the Veridata Managed Server.

Syntax

```
veridata.cli.server.listenAddress host
```

Example

```
veridata.cli.server.listenAddress host.example.com
```

Default Value

The default name of the managed server is `localhost`.

veridata.cli.server.timeout.seconds

This parameter specifies the time period (in seconds) Veridata CLI should wait for the JMX Server to respond to a CLI request.

Syntax

```
veridata.cli.server.timeout.seconds seconds
```

Example

```
veridata.cli.server.timeout.seconds 90
```

Default Value

The default time-out is `60` seconds.

Parameters for Report File Encryption

This section defines the configurable parameters used for report file encryption:

- server.encryption
- server.encryption.bits

server.encryption

When this parameter is set to true, the comparison report artifacts will be encrypted. Otherwise, the report contents will be in clear text.

Syntax

```
server.encryption=[true|false]
```

Example

```
server.encryption=false
```

Default Value

The default value is false.

server.encryption.bits

This parameter specifies the strength of the encryption algorithm. Valid values are 128, 192, and 256. If set to a value other than 128, you must install Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files.

Syntax

```
server.encryption.bits=[128|192|256]
```

Example

```
server.encryption.bits=128
```

Default Value

The default value is 128.

A

Moving from a Test to Production Environment

Here you can learn about the scripts that we provide for you to migrate your test system to the production environment.
This appendix includes the following sections:

- Moving Installations from a Source Environment to a Target Environment
- Additional Steps for Moving Oracle GoldenGate Veridata Repository

Moving Installations from a Source Environment to a Target Environment

Oracle Fusion Middleware provides various scripts that you use to move your environment.

To move Oracle Home that contains all binary files of your Veridata installation, use the `copyBinary` and `pasteBinary` scripts. After moving the Oracle Home, use the `copyConfig` and `pasteConfig` scripts to move a copy of the Veridata domain configuration including the Administration Server, Managed Server, and other components in the domain to the target environment.

> **Note:**
>
> Test to production migration is not supported for a compact domain or for repository databases other than Oracle and SQL Server.

Additional Steps for Moving Oracle GoldenGate Veridata Repository

In addition to the common procedures described in the guide , follow the instructions below for moving the Oracle GoldenGate Veridata repository to a target environment:

Moving Veridata Configuration Data from Test to Production

To export and import Veridata repository configuration data, use the scripts available in the `DOMAIN_HOME/veridata/bin` directory use the following steps:

1. To export the repository from the test environment, run the export script as follows:

   ```
   DOMAIN_HOME/veridata/bin/veridata_export.sh  -export /tmp/export.xml -all -
   wluser cmd_user -exportPassword
   ```

Table A-1 describes the arguments you specify while running the export and import scripts.

Table A-1 Arguments for the Veridata Test to Production Scripts

Argument	Description
`export`	Indicates that the script is exporting Veridata data from the test environment.
`import`	Indicates that the script is importing Veridata data to the production environment.
`wluser` *`cmd_user`*	Specifies the administrative WebLogic Server user.
`exportPassword`	Indicates that the script is exporting the *`cmd_user`* password.

2. Verify that any Veridata Agent host, port, and user password specifiers for the production host are updated in the just created `/tmp/export.xml` file.
3. If the Veridata Agent host or port has changed, then you must manually update the generated `export.xml` file with the new host and port informations for the Connections.
4. The export operation exports all of the data in the repository to XML file without validation. If your environment has some compare pairs for which the Validation Status is not successful, then the import operation will fail with the XML file generated by the export operation. To prevent failure of the import operation, you have the option to disable the compare pair validation. You can do this by updating the generated XML with these steps:
   a. Open the generated XML file.
   b. Search for the `configuration` tag. This tag has the `validation` attribute, which is set to `required`, by default.
   c. Set the `validation` attribute to one of the following values:

      `omit-failures` — Indicates that all successfully validated compare pairs will be added to the repository and other specified compare pairs will be ignored.

      `none` — Indicates that no compare pair validation is done before adding the compare pairs to the repository. You and then use the Veridata GUI to review and fix the validation problems. For example:

      Old tag: `<configuration operation="update" validation="required">`

      Updated tag: `<configuration operation="update" validation="omit-failures">`
5. To import the repository to the production environment, run the import script as follows:

   ```
   DOMAIN_HOME/veridata/bin/veridata_import.sh -update  /tmp/export.xml -wluser
   cmd_user
   ```

Applying Configuration Changes while Moving from Test to Production

While moving from a test to production environment, if there are any configuration changes for the Veridata Agent such as host and port changes *or* if there is any schema or catalog name changes in the compare pairs, you must first execute the following statements:

Task 1 For all databases

```
Update DEV_VERIDATA.TABLE_INFO set  SRC_SCHEMA_NAME = production_source_schema
Where    SRC_SCHEMA_NAME  = test_source_schema
 Update TABLE_INFO set  TARG_SCHEMA_NAME = production_target_schema Where
TARG_SCHEMA_NAME  = test_target_schema
```

Where *DEV_VERIDATA* is the name of the production repository schema.
Where *production_source_schema* is the name of the production source schema and *production_target_schema* is the name of the production target schema.
Where *test_source_schema* is the name of the test source schema and *test_target_schema* is the name of the test target schema.

Task 2 Appropriate to your database

For SQL Server and Sybase databases:

```
Update TABLE_INFO set  SRC_CATALOG_NAME = production_source_catalog Where
SRC_CATALOG_NAME  = test_source_catalog

Update TABLE_INFO set  TARG_CATALOG_NAME = production_target_catalog Where
TARG_CATALOG_NAME  = test_target_catalog
```

Where *production_source_catalog* is the name of the production source catalog and *production_target_catalog* is the name of the production target catalog.
Where *test_source_catalog* is the name of the test source catalog and *test_target _catalog* is the name of the test target catalog.

For NSK:
Update the table names in the `COMPARE_PAIR` table to replace the test node names and disk volume names with the production names using one of the following appropriate for your database:

- **For Oracle:**

```
Update COMPARE_PAIRS SET  SRC_TABLE_NAME = '\SPROD.$PDATA' ||
SUBSTR(SRC_TABLE_NAME, 12) Where SRC_TABLE_NAME like '\STEST.TDATA%'
Update COMPARE_PAIRS SET  TARG_TABLE_NAME = '\TPROD.$PDATA' ||
    SUBSTR(TARG_TABLE_NAME, 12) Where TARG_TABLE_NAME like '\TTEST.TDATA%'
```

- **For SQL Server:**

```
Update COMPARE_PAIRS SET  SRC_TABLE_NAME = '\SPROD.$PDATA' +
SUBSTRING(SRC_TABLE_NAME, 12, LEN(SRC_TABLE_NAME) -  12) Where SRC_TABLE_
    NAME like '\STEST.TDATA%'
Update COMPARE_PAIRS SET  TARG_TABLE_NAME = '\TPROD.$PDATA' +
SUBSTRING(TARG_TABLE_NAME, 12, LEN(TARG_TABLE_NAME) -12) Where TARG_TABLE_
    NAME like '\TTEST.TDATA%'
```

Modifying the Agent details in the Production Environment

Update the Veridata Agent details in the `CONNECTIONS` table of the production environment host as described below:

- If only the Agent host name needs to be changed, update the database as follows:

```
Update CONNECTIONS set MGR_NAME = 'prod host' where MGR_NAME 'test host'
```

- If there are more changes to the Veridata Agent, such as changes to the port number, User ID, password, and Repair User ID, then you should start the

Veridata application and update the environment using the UI or command-line tool.

For example, create an `/tmp/con.xml` XML file as follows:

```
#Create xml as below by filling placeholders between @ and connections can be
more than one.
    <?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE configuration SYSTEM
    "http://@VeridataServerHost@:@veridataServerPort@/veridata/
configuration.dtd">
    <configuration validation="required">
    <connection name="@conneciotnName@" host="@sqlManagerHostSrc@"
        port="@sqlManagerPortSrc@" user="@sqlConn0User@"
        password="@sqlConn0Password@" repairUser="@repairUsername@"
        repairPassword="@repairPassword@" agent-timeout="4000"
        truncate-spaces="false" fetch-size="3" use-ssl="false">
           <description>
            <![CDATA[
                SQL Scripting Source Connection
            ]]>
            </description>
   </connection>
   </configuration>
```

Update the Veridata Agent with your XML file using:

```
DOMAIN_HOME/veridata/bin/veridata_import.sh -update /tmp/con.xml -wlUser cmd_user
```

Start the Veridata Agent after making these changes.

B

Sample Configuration File

Here you can find the sample configuration file for using with the Oracle GoldenGate Veridata import and export utilities.
For more information about the parameters used in this configuration file, see Configuration File Element Reference.

Sample Configuration File

This section shows the contents of a sample configuration file. For more details about each element in this configuration file, see Configuration File Element Reference.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright (c) 2011, Oracle and/or its affiliates. All rights reserved. -->
<!DOCTYPE configuration SYSTEM "configuration.dtd">
<configuration validation="required">
    <connection name="sqlScriptingSource" host="localhost" port="7860"
        user="source2" password="source2" agent-timeout="6000" truncate-
spaces="true" fetch-size="4"/>
    <connection name="sqlScriptingTarget" host="localhost" port="7862"
        user="target2" password="target2"/>
    <connection name="connection-with-properties" host="localhost"
        port="7860" user="source2" password="source2" repairUser="source2"
repairPassword="source2" agent-timeout="4000"
        truncate-spaces="false" fetch-size="3" use-ssl="false">
        <description>
            <![CDATA[ SQL Scripting Source Connection with user defined properties]]>
        </description>
        <conn-properties datatype-name="array" format="clob"/>
        <conn-properties datatype-name="binary_double" format="number" scale="3"/>
        <conn-properties datatype-name="binary_float" format="dec_float"
precision="5"/>
        <conn-properties datatype-name="timestamp" format="binary_timestamp"
scale="10" timezone="(UTC+05:30) Kolkata - India Time (IT)"/>
    </connection>
    <connection name="nskScriptingSource" host="gg-xxxx.us.company.com" port="9999"/>
    <connection name="nskScriptingTarget" host="gg-xxxx.us.company.com""
port="9999" />


    <group name="column-mapping" source-conn="sqlScriptingSource" target-
conn="sqlScriptingTarget"
         source-catalog="" target-catalog="" source-schema="SOURCE2" target-
schema="TARGET2">
         <description>
            <![CDATA[
                This group has various types of column mapping specifications.
            ]]>
            </description>
        <compare-pair source-table="SYSMAPPING1" name="exlcudeCol6">
            <excluded-column name="COL6"/>
         </compare-pair>
        <compare-pair source-table="SYSMAPPING2" name="userDefinedKeys">
```

```
            <key-column source-name="COL1" target-name="COL2"/>
            <key-column source-name="COL2" target-name="COL3"/>
        </compare-pair>
        <compare-pair source-table="SYSMAPPING3" name="userDefinedColsWildCard">
            <column source-name="COL.*" />
        </compare-pair>
        <compare-pair source-table="SYSMAPPING3" name="userDefinedColsLiteral">
            <column source-name="COL5" target-name="COL6" type="literal"/>
            <column source-name="COL.*" />
        </compare-pair>
    </group>

    <group name="table-mapping" source-conn="sqlScriptingSource" target-
conn="sqlScriptingTarget"
         source-catalog="" target-catalog="">
         <description>
            <![CDATA[
                 This group has table mapping specifications.
            ]]>
          </description>
        <compare-pair source-table="SYSMAPPING1" target-table="SYSMAPPING1"
name="sameTables"
            source-schema="SOURCE2" target-schema="TARGET2" >
        </compare-pair>
        <compare-pair source-table="SYSMAPPING2" target-table="SYSMAPPING3"
name="diffTables"
            source-catalog="" target-catalog="" source-schema="SOURCE2" target-
schema="TARGET2">
        </compare-pair>
        <compare-pair source-table="CHAR_*" target-table="*"
name="sameTables"
            source-schema="SOURCE2" target-schema="TARGET2" >
        </compare-pair>
    </group>

    <group name="delta-processing" source-conn="sqlScriptingSource" target-
conn="sqlScriptingTarget"
        source-catalog="" target-catalog=""   source-schema="SOURCE2" target-
schema="TARGET2">
        <description>
            <![CDATA[
                 This group has delta processing specifications.
            ]]>
        </description>
        <compare-pair source-table="SYSMAPPING1" target-table="SYSMAPPING1"
name="scriptingConfigTest1"
            delta-processing="true" >
            <key-column source-name="COL1" target-name="COL2"/>
            <key-column source-name="COL2" target-name="COL3"/>
            <column source-name="COL5" target-name="COL6" type="literal"/>
            <delta-config>
                <source-config column-name="COL1">
                    <query><![CDATA[
                        SELECT MAX(COL1) from SYSMAPPING1
                        ]]>
                    </query>
                </source-config>
                <target-config column-name="COL2">
                    <query><![CDATA[
                        SELECT MAX(COL1) from SYSMAPPING1
                            ]]>
```

```
                    </query>
                </target-config>
            </delta-config>
        </compare-pair>
    </group>

    <group name="enscribe-partition" source-conn="SourceNSKConnection" target-
conn="TargetNSKConnection" validation="none">
         <description>
            <![CDATA[
                This group has all the tables for NSK
            ]]>
            </description>

        <compare-pair  source-file-pattern="\ZEUS.$FSS02.FSSVSRC.*" source-
table="ACCTN*"  target-file-pattern="\ZEUS.$FSS03.FSSVTAR.*" target-table="*" >
            <enscribe-key
                name = "Part1"
                end-key ="1000"
                format  ="hexadecimal"
                default ="false"
                side="source"/>
            <enscribe-key
                name = "Part1"
                start-key ="001"
                format  ="hexadecimal"
                default ="false"
                side="target"/>
            <enscribe-key
                name = "Both"
                start-key ="001"
                end-key ="1000"
                default ="true"/>
        </compare-pair>
    </group>

    <group name="sql-partition" source-conn="sqlScriptingSource" target-
conn="sqlScriptingTarget"
        source-catalog="" target-catalog=""  source-schema="SOURCE2" target-
schema="TARGET2">
        <description>
            <![CDATA[
                This group has sql partition specification.
            ]]>
        </description>

        <compare-pair source-table="SYSMAPPING1" name="PART1">
            <sql-partition name="partition_wo_default" >
                <![CDATA[
                    col4 > 50
                ]]>
            </sql-partition>
            <sql-partition name="part2" side="source">
                <![CDATA[
                    col2 > 20
                ]]>
            </sql-partition>
            <sql-partition name="part2" side="target">
                <![CDATA[
                    col3 > 30
                ]]>
```

```
            </sql-partition>
        </compare-pair>

        <compare-pair source-table="SYSMAPPING2" name="PART2">
            <sql-partition name="partition_default" default="true" >
                <![CDATA[
                    col3 > 30
                ]]>
            </sql-partition>
        </compare-pair>
    </group>

    <group name="compare-pair-with-pkey" source-conn="sqlScriptingSource" target-
conn="sqlScriptingTarget"
         source-catalog="" target-catalog="" source-schema="SOURCE2" target-
schema="TARGET2">
        <description>
            <![CDATA[
                This group has all the SYSMAPPING tables.
                SYSMAPPING3 uses the user defined index B_SYSMAPPING4_IDX.
            ]]>
        </description>

        <compare-pair source-table="SYSMAPPING3" source-pkey="B_SYSMAPPING3_IDX"/
>
        <compare-pair source-table="SYSMAPPING*" target-table="*">
        </compare-pair>
        <compare-pair source-table="SYSMAPPING5" exclude="true"/>
    </group>

    <group name="enscribe-expand-ddl" source-conn="SourceNSKConnection" target-
conn="TargetNSKConnection" validation="none">
         <description>
            <![CDATA[
                This group has expand ddl specification for NSK
            ]]>
            </description>
        <compare-pair  source-file-pattern="\ZEUS.$FSS02.FSSVSRC.*" source-
table="TELLER"  target-file-pattern="\ZEUS.$FSS03.FSSVTAR.*"
            target-table="TELLER" name="excludeCompKeyCol">
            <enscribe-info side="source"
                dictionary="$FSS04.FSSVDCT" record="ENSSRC-REC">
                <expandddl
                    redefined-columns ="include"
                    resolvedups="appendAlphaIndex"
                    expandGroupArrays="false"
                    ddl-separator="underscore"
                    zero-fill-length="1"
                    fix-long-names="false"
                    max-col-name-len="110"/>
            </enscribe-info>
            <enscribe-info side="target"
                dictionary="$FSS04.FSSVDCT" record="ENSSRC-REC">
                <expandddl />
                </enscribe-info>
            <excluded-column name="ENSCRIBE-NUMBER"/>
        </compare-pair>

        <compare-pair  source-file-pattern="\ZEUS.$FSS02.FSSVSRC.*" source-
table="TELLER"  target-file-pattern="\ZEUS.$FSS03.FSSVTAR.*"
            target-table="TELLER" name="userDefined">
```

```
            <enscribe-info side="source"
                dictionary="$FSS04.FSSVDCT" record="ENSSRC-REC"/>
            <enscribe-info side="target"
                dictionary="$FSS04.FSSVDCT" record="ENSSRC-REC"/>
            <key-column source-name="KEY1" target-name="KEY1"/>
            <column source-name="ENSCRIBE-STRING" target-name="ENSCRIBE-STRING"/>
            <column source-name="FIRST-NAME" target-name="FIRST-NAME"/>
            <column source-name="LAST-NAME" target-name="LAST-NAME"/>
            <column source-name="ENSCRIBE-NUMBER" target-name="ENSCRIBE-NUMBER"/
>
        </compare-pair>
    </group>

    <group name="include-exclude-filter" source-conn="sqlScriptingSource" target-
conn="sqlScriptingTarget"
         source-catalog="" target-catalog=""
            source-schema="SOURCE2" target-schema="TARGET2">
         <description>
            <![CDATA[
                This group has include/exclude filter description
            ]]>
         </description>

        <filter type="include" table="SYSMAPPING*">
            <colfilter type ="exclude">
                <colfiltercol name="COL3" />
                <colfiltercol name="COL6" />
            </colfilter>
        </filter>
        <filter type="exclude" table="SYSMAPPING4">
        </filter>
        <compare-pair source-table="SYSMAPPING1" target-table="*"
name="userDefinedCols"> <!-- exclude col6 -->
            <column source-name="COL5" target-name="COL5"/>
            <column source-name="COL6" target-name="COL6"/>
        </compare-pair>
        <compare-pair source-table="SYSMAPPING2" name="userDefinedKeys"> <!--
exclude col3 -->
            <key-column source-name="COL1" target-name="COL2"/>
            <key-column source-name="COL2" target-name="COL3"/>
        </compare-pair>
        <compare-pair source-table="SYSMAPPING3" target-table="*"><!-- exclude col3,
col6 -->
        </compare-pair>
        <compare-pair source-table="SYSMAPPING4" target-table="*"/>

    </group>

    <group name="quotedSchemaQuotedTable" source-conn="sqlScriptingSource" target-
conn="sqlScriptingTarget"
         source-catalog="" target-catalog=""
            source-schema="&amp;quot;abc 11&amp;quot;" target-schema="&amp;quot;abc 11&amp;quot;">
         <description>
            <![CDATA[
                SQL group with simple quoted schema and quoted table name
            ]]>
          </description>
        <compare-pair source-table="&amp;quot;Quoted Table&amp;quot;" target-table="*"  />
        <compare-pair source-table="&amp;quot;Quoted*Table&amp;quot;" target-table="*"  />
    </group>
```

```
    <group name="group-schema-wildcard" source-conn="sqlScriptingSource" target-
conn="sqlScriptingTarget"
         source-catalog="" target-catalog="" source-schema="source*" target-
schema="*">
        <description>
            <![CDATA[
                SQL group with source-schema wildcard at group level and no compare-
pair schema.
            ]]>
        </description>
        <filter type="include" table="CHAR_TYPES*">
        </filter>
        <filter type="exclude" table="CHAR_TYPES2*">
        </filter>
        <compare-pair source-table="CHAR_TYPE*" target-table="*">
        </compare-pair>
    </group>


    <job name="test" profile="allParams">
        <group name="column-mapping"/>
        <group name="table-mapping"/>
        <group name="delta-processing"/>
    </job>

    <profile name="allParams">
        <description>
            <![CDATA[
                Full Profile description.
            ]]>
        </description>
        <profile-general>
            <param name="oos-format" value="xml" />
            <param name="oos-xml-chunk-size" value="1000" />
            <param name="reports-insync" value="true" />
            <param name="reports-inflight" value="true" />
        </profile-general>
        <sorting-method>
            <param name="sort-method" value="server" />
            <param name="sort-src-temp-dir" value="/dummy/location" />
            <param name="sort-tar-temp-dir" value="/dummy/location" />
        </sorting-method>
        <initial-compare>
            <param name="max-thread" value="6" />
            <param name="max-oos-record" value="777777" />
            <param name="output-oos-rpt" value="true" />
            <param name="update-rpt-second" value="100" />
            <param name="update-rpt-record" value="100" />
            <param name="limit-input-row" value="100" />
            <param name="src-oracle-hint" value="FIRST_ROWS(10)" />
            <param name="tar-oracle-hint" value="FIRST_ROWS(10)" />

            <param name="rpt-msg" value="both" />
            <param name="rpt-warn-msg-threshold" value="100" />

            <param name="src-agent-static-port" value="777" />
            <param name="tar-agent-static-port" value="777" />

            <param name="src-nsk-name" value="$AA*" />
            <param name="src-nsk-cpu" value="2" />
            <param name="src-nsk-priority" value="1" />
```

```
            <param name="tar-nsk-name" value="$AA*" />
            <param name="tar-nsk-cpu" value="2" />
            <param name="tar-nsk-priority" value="1" />
        </initial-compare>
        <confirm-out-of-sync>
            <param name="coos-enable" value="false" />
            <param name="coos-concurrent" value="false" />
            <param name="batch-size" value="15"/>
            <param name="coos-delay" value="2" />
            <param name="max-oos-record" value="777777" />
            <param name="output-oos-rpt" value="true" />
            <param name="update-rpt-second" value="100" />
            <param name="update-rpt-record" value="100" />
            <param name="src-oracle-hint" value="FIRST_ROWS(10)" />
            <param name="tar-oracle-hint" value="FIRST_ROWS(10)" />

            <param name="rpt-msg" value="both" />
            <param name="rpt-warn-msg-threshold" value="100" />

            <param name="src-agent-static-port" value="777" />
            <param name="tar-agent-static-port" value="777" />

            <param name="src-nsk-name" value="$AA*" />
            <param name="src-nsk-cpu" value="2" />
            <param name="src-nsk-priority" value="1" />
            <param name="tar-nsk-name" value="$AA*" />
            <param name="tar-nsk-cpu" value="2" />
            <param name="tar-nsk-priority" value="1" />
        </confirm-out-of-sync>
        <repair>
            <param name="repair-after-compare" value="true" />
            <param name="batch-size" value="15" />
            <param name="txn-size" value="2" />
            <param name="concurrent-operation" value="2" />
            <param name="check-change-value" value="false" />
            <param name="terminate-max-warn" value="77777" />
            <param name="write-success-rpt" value="false" />
            <param name="disable-trigger" value="true" />
        </repair>
    </profile>
</configuration>
```

C

Profile Parameters

Here you can find the profile parameter decryption for you to configure the profiles used with the Oracle GoldenGate Veridata import and export utilities.
For more information about the parameters used in this configuration file, see Configuration File Element Reference.

General (profile-general)

Name on UI	Name to be used in configuration XML file	Default Values	Allowed Values	Type
Out-Of-Sync Output Format	oos-format	binary	binary, xml, both, none	Enumeration
Maximum Size of Each Out-Of-Sync XML Chunk (Rows)	oos-xml-chunk-size	500	1 to 100000	int
Report in-sync rows to report file	reports-insync	false	true, false	boolean
Report in-sync after in-flight rows to report file	reports-inflight	false	true, false	boolean

Sorting Method (sorting-method)

Name on UI	Name to be used in configuration XML file	Default Values	Allowed Values	Type
Sort Data Using	sort-method	database	server, database	Enumeration
Maximum Memory Usage (MB)	sort-max-memory	50	1 to 100000	int
Temporary Storage Directory for Source Data	sort-src-temp-dir	string		
Temporary Storage Directory for Target Data	sort-tar-temp-dir	string		

Initial Compare (initial-compare

Name on UI	Name to be used in config XML file	Default Values	Allowed Values	Type
Max Concurrent Comparison Threads	max-thread	4	1 to 20	Int
Terminate when Maximum Records Out-Of-Sync	max-oos-record	100000	0 to 100000000	int
Output Out-Of-Sync Record Details to Report File	output-oos-rpt	false	true, false	boolean
Update Report file Every (seconds)	update-rpt-second	0	0 to 1000000	int
Update Report file Every (records)	update-rpt-record	0	0 to 100000000	int
Limit Number of Input Rows	limit-input-row	0	0 to 100000000	int
Source Oracle optimizer hint	src-oracle-hint	string		
Target Oracle optimizer hint	tar-oracle-hint	string		
Generate Messages	rpt-msg	none	none, info, warning, both	Enumeration
Generate Warning Messages For Out-Of-Sync Rows After (differences)	rpt-warn-msg-threshold	50	0 to 1000000	int
Use Static Listening Port For Agent During Row Hash On Source (0 to use dynamic port list)	src-agent-static-port	0	0 to 65535	int
Use Static Listening Port For Agent During Row Hash On Target (0 to use dynamic port list)	tar-agent-static-port	0	0 to 65535	int

Name on UI	Name to be used in config XML file	Default Values	Allowed Values	Type
Source Process Name Starting With (Must start with '$', followed by two letters, and end with '*'. Example: $AA*)	src-nsk-name	string		
Source Process CPU Number	src-nsk-cpu	-1	-1 to 16	int
Source Process Priority	src-nsk-priority	0	0 to 1000000	int
Target Process Name Starting With (Must start with '$', followed by two letters, and end with '*'. Example: $AA*)	tar-nsk-name	string		
Target Process CPU Number	tar-nsk-cpu	-1	-1 to 16	int
Target Process Priority	tar-nsk-priority	0	0 to 1000000	int

Confirm-Out-Of-Sync (confirm-out-of-sync)

Name on UI	Name to be used in configuration XML file	Default Values	Allowed Values	Type
Perform Confirm Out-Of-Sync Step	coos-enable	true	true, false	boolean
Run Concurrently With Initial Compare	coos-concurrent	true	true, false	boolean
Confirm-Out-Of-Sync Batch Size	Batch-size	10	1 to 100	int
Delay Confirm-Out-Of-Sync By (seconds)	coos-delay	0	0 to 1000000	int
Terminate when Maximum Records Out-Of-Sync	coos-max-oos	100000	0 to 100000000	int
Output Out-Of-Sync Record Details to Report File	coos-output-oos-rpt	false	true, false	boolean

Name on UI	Name to be used in configuration XML file	Default Values	Allowed Values	Type
Update Report file Every (seconds)	coos-output-rpt-second	0	0 to 100000000	int
Update Report file Every (records)	coos-output-rpt-record	0	0 to 100000000	int
Source Oracle optimizer hint	coos-src-oracle-hint	string		
Target Oracle optimizer hint	coos-tar-oracle-hint	string		
Generate Messages	rpt-msg	none	none, info, warning, both	Enumeration
Generate Warning Messages For Out-Of-Sync Rows After (differences)	rpt-warn-msg-threshold	50	0 to 1000000	int
Use Static Listening Port For Agent During Row Hash On Source (0 to use dynamic port list)	src-agent-static-port	0	0 to 65535	int
Use Static Listening Port For Agent During Row Hash On Target (0 to use dynamic port list)	tar-agent-static-port	0	0 to 65535	int
Source Process Name Starting With (Must start with '$', followed by two letters, and end with '*'. Example: $AA*)	src-nsk-name	string		
Source Process CPU Number	src-nsk-cpu	-1	-1 to 16	int
Source Process Priority	src-nsk-priority	0	0 to 1000000	int
Target Process Name Starting With (Must start with '$', followed by two letters, and end with '*'. Example: $AA*)	tar-nsk-name	string		

Name on UI	Name to be used in configuration XML file	Default Values	Allowed Values	Type
Target Process CPU Number	tar-nsk-cpu	-1	-1 to 16	int
Target Process Priority	tar-nsk-priority	0	0 to 1000000	int

Repair (repair)

Name on UI	Name to be used in configuration XML file	Default Values	Allowed Values	Type
Run Repair Automatically after Compare	repair-after-compare	False	True,false	boolean
Repair batch size	batch-size	10	1 to 100	int
Repair transaction size	txn-size	1	0 to 100	int
Concurrent Repair Operations	concurrent-operation	1	1 to 100	int
Check changed values	check-change-value	true	true, false	boolean
Terminate when maximum repair warnings	terminate-max-warn	10000	0 to 2147483647	Int
Write Repair Success Messages to Report	write-success-rpt	true	true, false	boolean
Disable DB Triggers Session Based	disable-trigger	false	true, false	boolean
Report				
Disable DB Triggers Session Based	disable-trigger	false	true, false	boolean

Index