This guide explains how to run and administer the Oracle GoldenGate Veridata data comparison solution.
Contents

Preface ................................................................................................................................................................ vii

1 Introduction to Oracle GoldenGate Veridata
   1.1 Oracle GoldenGate Veridata Architecture .............................................................................. 1-1
   1.2 Configuring Single Sign-on for Oracle GoldenGate Veridata .............................................. 1-3
   1.3 Comparing Data with Oracle GoldenGate Veridata ............................................................ 1-4
       1.3.1 Oracle GoldenGate Veridata Comparison Objects ........................................................ 1-4
       1.3.2 Satisfying Uniqueness Requirements ............................................................................ 1-5
       1.3.3 How Oracle GoldenGate Veridata Compares Data .................................................... 1-5
   1.4 Viewing Comparison Results .................................................................................................... 1-6
       1.4.1 Out-of-Sync Report ........................................................................................................ 1-6
       1.4.2 Comparison Report ....................................................................................................... 1-7

2 Configuring Oracle GoldenGate Veridata Security
   2.1 Oracle GoldenGate Veridata Security Overview ................................................................... 2-1
   2.2 Configuring an SSL Connection between Veridata Server and Veridata Agents ............... 2-1
       2.2.1 One-Way and Two-Way SSL Connections ................................................................. 2-2
       2.2.2 Enabling SSL: Main Steps ......................................................................................... 2-2
       2.2.3 SSL Settings for Oracle GoldenGate Veridata Agent ............................................... 2-3
       2.2.4 SSL Settings for GoldenGate Veridata Server .......................................................... 2-4
       2.2.5 Creating Keystores and Self-Signed Certificates by using the Keytool Utility .......... 2-4
       2.2.6 Using OPSS Keystore Service to Manage Veridata Keystores ................................ 2-8
       2.2.7 Modifying the Veridata Agent Wallet ......................................................................... 2-8
   2.3 Securing the Oracle GoldenGate Veridata Files .................................................................... 2-9
       2.3.1 Controlling Access to the Installation Directories ...................................................... 2-9
       2.3.2 Securing files that Contain Business Data .................................................................. 2-9
   2.4 Securing Access to Oracle GoldenGate Veridata by Defining User Roles ....................... 2-9
   2.5 Changing Database Schema Passwords .............................................................................. 2-12
   2.6 Encrypting Report Files ......................................................................................................... 2-13
       2.6.1 Enabling Report Encryption ....................................................................................... 2-13
       2.6.2 Using the reportutil Tool to view Reports ................................................................. 2-14

3 Running the Oracle GoldenGate Veridata Programs
   3.1 Starting and Stopping the C-agent and Manager ................................................................. 3-1
   3.2 Starting and Stopping the Java-Based Components ............................................................. 3-1
4 Running Comparisons from the Command Line

4.1 Overview of the vericom Tool ................................................................. 4-1
4.2 Running Vericom .................................................................................... 4-2
4.3 Vericom exit statuses ........................................................................... 4-6
4.4 Vericom Output Examples ..................................................................... 4-6

5 Using the Veridata Import and Export Utilities

5.1 Introduction to the Import and Export Utilities ..................................... 5-1
5.1.1 Supported Configurations ................................................................. 5-1
5.2 Running the Import and Export Utilities ............................................. 5-2
5.2.1 Using the Export Utility ................................................................. 5-2
5.2.2 Using the Import Utility ................................................................. 5-3
5.2.3 Processing the Configuration ........................................................ 5-3
5.3 Configuration File Element Reference ................................................ 5-5
configuration ......................................................................................... 5-6
column ................................................................................................. 5-8
colfilter ............................................................................................... 5-9
colfiltercol ......................................................................................... 5-10
compare-pair ....................................................................................... 5-11
connection ......................................................................................... 5-14
conn-properties .................................................................................. 5-16
delta-config ......................................................................................... 5-17
description ........................................................................................ 5-18
enscribe-info ....................................................................................... 5-19
enscribe-key ....................................................................................... 5-20
excluded-column .............................................................................. 5-21
expandddl .......................................................................................... 5-22
filter .................................................................................................. 5-23
group ................................................................................................. 5-24
job ..................................................................................................... 5-26
profile ............................................................................................... 5-27
key-column ....................................................................................... 5-28
profile-general .................................................................................. 5-29
sorting-method ................................................................................. 5-30
initial-compare .................................................................................. 5-31
confirm-out-of-sync .......................................................................... 5-32
param ............................................................................................... 5-33
repair .............................................................................................. 5-34
sql-partition ...................................................................................... 5-35
6 Running Veridata GoldenGate Parameter Processing
   6.1 Overview of the Command Line Interface ................................................................. 6-1
   6.2 Running VGPP .............................................................................................................. 6-2
   6.2.1 Using a Property File .............................................................................................. 6-3
   6.3 Parameter Handling ................................................................................................... 6-4
   6.4 Map and Table Statement Handling ......................................................................... 6-5

7 Oracle GoldenGate Veridata Server Configuration Parameters
   7.1 Overview of the Server Memory ................................................................................ 7-1
   7.2 Estimating Memory Usage ....................................................................................... 7-2
   7.3 How to Set a Parameter ........................................................................................... 7-2
   7.4 Parameter Descriptions ........................................................................................... 7-2
     Server Parameters ....................................................................................................... 7-4
     server.veridata_data .......................................................................................................... 7-5
     server.persistence_db_type ............................................................................................ 7-6
     server.meta_session_handle_timeout .......................................................................... 7-7
     server.max_concurrent_jobs ......................................................................................... 7-8
     server.max_concurrent_comparison_threads .................................................................. 7-9
     server.mapped_sort_buffers ......................................................................................... 7-10
     server.max_sort_memory ............................................................................................. 7-11
     server.concurrent.writers .......................................................................................... 7-12
     server.concurrent.readers .......................................................................................... 7-13
     server.number_sort_threads ....................................................................................... 7-14
     Parameters for Configuring SSL Communication ...................................................... 7-15
     server.useSsl .................................................................................................................. 7-16
     server.ssl.client.allowTrustedExpiredCertificates .................................................... 7-17
     server.ssl.client.identitystore.keyfactory.alg.name .................................................. 7-18
     server.ssl.client.truststore.keyfactory.alg.name ....................................................... 7-19
     server.ssl.algorithm.name ........................................................................................... 7-20
     Parameters for Veridata Command-Line Utility .......................................................... 7-21
     veridata.cli.run_from_managed_server ....................................................................... 7-22
     veridata.cli.managed_server_name ............................................................................. 7-23
     veridata.cli.server.listenAddress ............................................................................... 7-24
     veridata.cli.server.timeout.seconds ......................................................................... 7-25
     Parameters for Report File Encryption ........................................................................ 7-26
     server.encryption .......................................................................................................... 7-27
     server.encryption.bits ................................................................................................. 7-28

A Moving from a Test to Production Environment
   A.1 Moving Installations from a Source Environment to a Target Environment ............ A-1
   A.2 Additional Steps for Moving Oracle GoldenGate Veridata Repository .................... A-1
   A.2.1 Moving Veridata Configuration Data from Test to Production .............................. A-1
A.2.2 Applying Configuration Changes while Moving from Test to Production .......... A-2
A.2.3 Modifying the Agent details in the Production Environment ......................... A-3

B Sample Configuration File
B.1 Sample Configuration File .................................................................................. B-1

C Profile Parameters
C.1 General (profile-general) ..................................................................................... C-1
C.2 Sorting Method (sorting-method) ......................................................................... C-1
C.3 Initial Compare (initial-compare) ......................................................................... C-2
C.4 Confirm-Out-Of-Sync (confirm-out-of-sync) ......................................................... C-3
C.5 Repair (repair) ...................................................................................................... C-4
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.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

The complete Oracle GoldenGate documentation set includes the following components:

- Release Notes for Oracle GoldenGate Veridata
- Installing and Configuring Oracle GoldenGate Veridata
- Upgrading Oracle GoldenGate Veridata
- User’s Guide

Conventions

The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>boldface</td>
<td>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.</td>
</tr>
<tr>
<td>Convention</td>
<td>Meaning</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
</tr>
<tr>
<td>italic, italic</td>
<td>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.</td>
</tr>
<tr>
<td>MONOSPACE, monospace</td>
<td>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.</td>
</tr>
<tr>
<td>UPPERCASE</td>
<td>Uppercase in the regular text font indicates the name of a utility unless the name is intended to be a specific case.</td>
</tr>
<tr>
<td>{ }</td>
<td>Braces within syntax enclose a set of options that are separated by pipe symbols, one of which must be selected, for example: (option1</td>
</tr>
<tr>
<td>[ ]</td>
<td>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</td>
</tr>
</tbody>
</table>
This chapter describes how to use Oracle GoldenGate Veridata. It provides an overview of roles and interactions of the components, how to configure components, and how Oracle GoldenGate Veridata compares tables and repairs out-of-sync tables.

This chapter includes the following sections:

- Oracle GoldenGate Veridata Architecture
- Configuring Single Sign-on for Oracle GoldenGate Veridata
- Comparing Data with Oracle GoldenGate Veridata
- Viewing Comparison Results

### 1.1 Oracle GoldenGate Veridata Architecture

Oracle GoldenGate Veridata compares one set of data to another and identifies data that is out-of-sync, and allows you to repair any data that is found out-of-sync. Oracle GoldenGate Veridata supports high-volume, 24x7 heterogenous 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 GoldenGate Veridata can run concurrently with data transactions and replication, while still producing an accurate comparison report.

Oracle GoldenGate Veridata will map column data types across different types of databases automatically, or you can map columns manually in cases where the automatic mapping is not sufficient to accommodate format differences in a heterogeneous environment. For detailed information about this feature in Veridata Web User Interface, see Oracle GoldenGate Veridata Online Help. Alternatively, you can map data by manually uploading an XML file using the veridata_import utility. For more information, see Chapter 5, “Using the Veridata Import and Export Utilities”.

For the purposes of this documentation, the following terms are considered synonymous:

- tables and files
- columns and fields
- rows and records
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
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 Veridata repository. This data is stored in files on the file system of the Veridata Server.
**Oracle GoldenGate Veridata Agent**
The Oracle GoldenGate Veridata 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 detail for out-of-sync rows

**Oracle GoldenGate Veridata Manager**
The Manager process is part of the C-code based agent that is required for the NonStop platform. It controls the Oracle GoldenGate Veridata 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>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vericom</td>
<td>Enables you to run comparisons by using automated programs. See Chapter 4, “Running Comparisons from the Command Line”.</td>
</tr>
<tr>
<td>veridata_import</td>
<td>Maps comparison objects and rules in an XML file and imports it into the repository. See Chapter 5, “Using the Veridata Import and Export Utilities”.</td>
</tr>
<tr>
<td>veridata_export</td>
<td>Maps comparison objects and rules in the repository and exports them to an XML file. See Chapter 5, &quot;Using the Veridata Import and Export Utilities&quot;.</td>
</tr>
<tr>
<td>veridata_param_process</td>
<td>The Veridata GoldenGate Parameter Processing (VGPP) command line tool to use Oracle GoldenGate parameter files. See Chapter 1, “Introduction to Oracle GoldenGate Veridata”.</td>
</tr>
<tr>
<td>reportutil</td>
<td>Supports viewing encrypted report files and out-of-sync data.</td>
</tr>
</tbody>
</table>

**1.2 Configuring Single Sign-on for Oracle GoldenGate Veridata**
Oracle GoldenGate Veridata 12c (12.2.1) supports Single Sign-on (SSO) mechanism for authentication. To configure SSO, you should set the SSO properties for Veridata Server and also configure the logout URL for the SSO session.

The web parameter `web.singleSignOutUrl` in the `DOMAIN_HOME/config/veridata/veridata.cfg` file is used to configure SSO for the Veridata Server. The parameter usage is explained in the `veridata.cfg` file:

```ini
# (web.singleSignOutUrl) as
# web.singleSignOutUrl - Specify the Single Sign Out URL here:
# Formats: /oamsso/logout.html?end_url=/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.
```
To configure SSO, set the `web.singleSignOutUrl` parameter and run the `configureVeridata` script as follows:

**To Configure SSO Logout**

```bash
DOMAIN_HOME/veridata/bin/configureVeridata.sh -pUweb.singleSignOutUrl=Single sign out URL
```

**To Reset SSO Logout**

If your domain is no longer using SSO, you can optionally remove the SSO logout configuration as follows:

```bash
DOMAIN_HOME/veridata/bin/configureVeridata.sh -pUweb.singleSignOutUrl=default
```

### 1.3 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.

#### 1.3.1 Oracle GoldenGate Veridata Comparison Objects

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

- **Configure data source connections:** Oracle GoldenGate Veridata Server connects to Oracle GoldenGate Veridata Agents that interact with the databases that contain the data that is to be compared. A connection is defined by a host, the port number of an Oracle GoldenGate Veridata Agent on that host, and the data source that is accessed by the agent.

- **Configure groups:** You need to configure at least one compare group that is linked to a set of source and target data source connections. A group is a logical container for organizing the objects to be compared.

- **Configure compare pairs:** You need to configure one or more compare pairs for each group that you create. A compare pair is a set of corresponding source and target tables or files. As part of configuring compare pairs, you map source and target columns to establish a structural relationship between the two objects.

- **Configure profiles:** A profile contains settings for run-time parameters and can be applied globally to a job, as well as to a specific compare pair as an override to the job profile. Profile parameters encompass considerations such as sorting method, thread and memory usage, reporting output, and so forth. Defining run profiles is optional, because Oracle GoldenGate Veridata includes a default profile that contains settings that apply to most usage scenarios. However, as you gain experience with Oracle GoldenGate Veridata, you may want to customize the default profile or create your own custom profiles.

- **Configure jobs:** A job is a logical container for one or more compare groups and is the unit of work by which comparison processing is executed. Within one or more jobs, you can manage and run large volumes of compare groups across numerous databases and systems, and you can control the timing of those comparisons.

For more information, see the *Oracle GoldenGate Veridata Online Help*. 
1.3.2 Satisfying Uniqueness Requirements

Oracle GoldenGate Veridata relies on some form of 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, Oracle GoldenGate Veridata uses the smallest unique index.
- **User-defined Key**: If a table or file has neither a primary nor unique key, you can define an existing index or set of columns for comparison purposes when defining a compare pair. However, although primary or unique keys can be mapped automatically, user-defined keys must be mapped manually. A user-defined key can also be used to override existing keys or indexes if you prefer a different ordering method.

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

1.3.3 How Oracle GoldenGate Veridata Compares Data

Comparison activities are divided into the following steps. You can change some of the aspects of these steps by making parameter changes in Oracle GoldenGate Veridata Web.

1.3.3.1 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, while still providing a highly reliable (but not absolute) and efficient mechanism for determining whether two rows contain the same or different column values.

For more assurance of discovering out-of-sync rows, you can configure Oracle GoldenGate Veridata to compare non-key rows column-for-column, instead of using a hash. 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 feature during the initial comparison step if you are using server-side sorting. Delta processing is a performance feature by which Oracle GoldenGate Veridata detects which data blocks in the database 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 regardless of whether they changed or not.

There are two steps to delta processing:

- Collecting the base modification time of the previous run for subsequent delta comparisons. This step is always done when delta processing is enabled for a compare pair.
- Comparing data that has been modified since the base comparison, using the information that was collected in the first step. This step is enabled by clicking the **Enable Delta Processing** button on the Compare Pair Configuration page and the Run/Execute Job page of Oracle GoldenGate Veridata Web. The **Disable Delta**
**Processing** button allows you to disable the delta comparison step in case there were modifications, such as table reorganizations, that can invalidate the collected delta base information.

For more information about delta processing, see "Using Delta Processing" in *Oracle GoldenGate Veridata User’s Guide* or the *Oracle GoldenGate Veridata 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 at this point the comparison is inconclusive. When replication is working concurrently with a comparison, especially if there is replication latency, rows can appear to be out-of-sync when, in fact, the current data is *in flight* (somewhere in the replication flow) and replication will soon synchronize them again.

### 1.3.3.2 Confirmation Step

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

- **in-flight**: the row was out-of-sync in the initial comparison step, but has since been updated. In this case, 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. Even a status of in-sync 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 has not been updated since the initial comparison step took place, and therefore 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 after a specified replication latency threshold has expired. 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.

### 1.4 Viewing Comparison Results

Upon completion of a job, you can view the comparison reports and the out-of-sync report by using *Oracle GoldenGate Veridata Web User Interface* or by viewing the files themselves.

If report encryption is enabled for the Veridata Server, you need to use the *reportutil* tool to view the report files. See Section 2.6, "Encrypting Report Files". The Veridata Web User Interface automatically decrypts the file before displaying them.

#### 1.4.1 Out-of-Sync Report

You have the option to store an out-of-sync 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 are used for viewing row differences using the Oracle GoldenGate Veridata Web Interface, and the report is also used to re-compare out-of-sync rows later. To re-compare rows, you select run options to execute another confirmation step, which compares the current state of just those rows and then reports which ones remain out-of-sync after replication or another restorative procedure has been applied.

■ **OOSXML file**: When stored as XML, the OOS report is written to an OOSXML file and is stored in a structured way that conforms to an internal XML schema. XML has many advantages, the largest being that it can be manipulated easily by many tools. In its XML form, the file contains all of the information, including metadata, that is needed to select rows for re-synchronization by external programs.

### 1.4.2 Comparison Report

Each finished job, group, and compare pair generates a comparison report. The report file contains details about the comparisons that were performed, such as:

- Comparison parameters used
- The number of rows compared and out-of-sync
- The timing of the comparison
- Performance statistics
- Source and target data values

The files themselves are stored as follows:

By default, the OOS files are located in sub-directories of the Oracle GoldenGate Veridata Server installation directory:

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

You can change the default location by specifying another path for the `server.veridata_data` property in the `veridata.cfg` file.

These directories are further organized by run ID, job name, group name, and compare pair. In the OOSXML directory, the files with the `.oosxml` extension 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 choose to encrypt the comparison reports. For more information, see Section 2.6, "Encrypting Report Files".
This chapter explains how to set security for Oracle GoldenGate Veridata. This chapter includes the following sections:

- Oracle GoldenGate Veridata Security Overview
- Configuring an SSL Connection between Veridata Server and Veridata Agents
- Securing the Oracle GoldenGate Veridata Files
- Securing Access to Oracle GoldenGate Veridata by Defining User Roles
- Changing Database Schema Passwords
- Encrypting Report Files

2.1 Oracle GoldenGate Veridata Security Overview

When using Oracle GoldenGate Veridata, you will be selecting, viewing and storing data values from the tables or files of your business applications. Care must be taken to protect access to the following components:

- The files, programs, and directories in the Oracle GoldenGate Veridata installation directories
- Data files that contain the results of data comparisons
- The Oracle GoldenGate Veridata Web User Interface, where data values can be viewed

2.2 Configuring an SSL Connection between Veridata Server and Veridata Agents

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

Note: The Veridata Agent for NonStop platforms do not support SSL communication.
In an SSL scenario, the Veridata Server is considered as the SSL Client and the Veridata Agents as the SSL Servers. The Veridata Server and Agents authenticate each other’s identity. The data exchanged between the server and agent is also encrypted.

2.2.1 One-Way and Two-Way SSL Connections

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

In one-way SSL connection, the SSL Client (Veridata Server) should trust the SSL Server (Veridata Agent). In two-way SSL, mutual trust is required between the SSL Server and the SSL Client. You can either use self-signed certificates or CA signed certificates to enable SSL.

Using self-signed certificates

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

- Create self-signed certificates for all Veridata Agents.
- Upload all Veridata Agent certificates to the VeridataWebTrustStore of the Veridata Server. See Section 2.2.6, "Using OPSS Keystore Service to Manage Veridata Keystores".

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

- Create self-signed certificates for all Veridata Agents.
- Upload all Veridata Agent certificates to the VeridataWebTrustStore of the Veridata Server. See Section 2.2.6, "Using OPSS Keystore Service to Manage Veridata Keystores".
- Create self-signed certificate for the identity store of the Veridata WebLogic Server.
- Upload the WebLogic Server identity certificate to all Veridata Agent truststores.

For more information about creating and importing certificates, see Section 2.2.5, "Creating Keystores and Self-Signed Certificates by using the Keytool Utility".

Using CA-signed certificates

To establish one-way SSL using CA signed certificates:

- Use certificates issued by the same Certificate Authority (CA) for all Veridata Agents.
- Trust the root CA certificate in the Veridata Weblogic Server.

To establish two-way SSL using CA signed certificates:

- Use certificates issued by the same Certificate Authority (CA) for all Veridata Agents.
- Trust the root CA certificate in the Veridata Weblogic Server.
- Use the certificate issued by a CA for identity store of Veridata Weblogic Server.
- Trust the root CA certificate used in the previous step in the Veridata agent truststore.

2.2.2 Enabling SSL: Main Steps

Oracle GoldenGate Veridata Server and Oracle GoldenGate Veridata Agents are not enabled for SSL by default. If you decide to use SSL, you must enable the properties for the server and the agents. See SSL Settings for Oracle GoldenGate Veridata Agent and SSL Settings for GoldenGate Veridata Server.
You must also create the identity and trust keystores. Create self-signed certificates if you are not using a Certificate Authority (CA) certificate. See Creating Keystores and Self-Signed Certificates by using the Keytool Utility.

To verify and establish an SSL connection between the Veridata Server and an Agent, follow these steps:


2. Restart the Veridata Web Server.

   a. Obtain the agent side keystores. See Using OPSS Keystore Service to Manage Veridata 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 Veridata Agent Wallet.

5. Start the agent.

6. If the trust is established properly between agent keystores and corresponding Veridata Server counterpart present in the OPSS Keystore Service, then SSL communication is established.

2.2.3 SSL Settings for Oracle GoldenGate Veridata Agent

By default, SSL is disabled for the Oracle GoldenGate Veridata Agent. To configure SSL, edit the following properties in the agent.properties file for your Veridata Agent.

<table>
<thead>
<tr>
<th>Table 2–1</th>
<th>SSL Parameters in agent.properties file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>server.useSsl</td>
<td>Enables or disables SSL Communication between the Veridata Agent and Veridata Server. Possible values are: true: Enables SSL communication false: Disables SSL communication</td>
</tr>
<tr>
<td>server.use2WaySsl</td>
<td>Specifies whether the SSL communication is one-way or two-way. Options are: true: Uses two-way SSL communication false: Uses one-way SSL communication</td>
</tr>
<tr>
<td>server.identitystore.type</td>
<td>Specifies the type of keystore used for SSL configuration.</td>
</tr>
<tr>
<td>server.identitystore.path</td>
<td>Specifies the path for the server identity keystore.</td>
</tr>
<tr>
<td>server.truststore.type</td>
<td>Specifies the type of truststore used for SSL configuration.</td>
</tr>
<tr>
<td>server.truststore.path</td>
<td>Specifies the path for the server truststore.</td>
</tr>
</tbody>
</table>
Configuring an SSL Connection between Veridata Server and Veridata Agents

2.2.4 SSL Settings for GoldenGate Veridata Server

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

You can also establish SSL communication only for certain connections. To do this, edit the connection properties in the Oracle GoldenGate Veridata web user interface. For more information, see the Oracle GoldenGate Veridata Online Help.

**Table 2–2 SSL Settings in veridata.cfg file**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>server.useSsl</td>
<td>Enables or disables SSL Communication between the Veridata Agent and Veridata Server. Possible values are: true: Enables SSL communication false: Disables SSL communication</td>
<td>false</td>
</tr>
<tr>
<td>server.ssl.client.identitystore.keyfactory.alg.name</td>
<td>Algorithm name of the keyfactory used for SSL server identity store.</td>
<td>SunX509</td>
</tr>
<tr>
<td>server.ssl.client.truststore.keyfactory.alg.name</td>
<td>Algorithm name of the keyfactory used for SSL server trust store.</td>
<td>SunX509</td>
</tr>
<tr>
<td>server.ssl.algorithm.name</td>
<td>SSL algorithm name. <strong>Note:</strong> This value of this parameter must be same for the Veridata Agent and Veridata Server.</td>
<td>TLS</td>
</tr>
</tbody>
</table>

2.2.5 Creating Keystores and Self-Signed Certificates by using the Keytool Utility

For mutual authentication and to establish SSL communication, the Veridata Server and the Veridata 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 Java documentation at http://docs.oracle.com/javase/7/docs/technotes/tools/#security.

2.2.5.1 Creating an Identity Keystore with a Self-Signed Certificate
The following keytool command creates a keystore containing a self-signed certificate:
```shell
keytool -genkey -keystore certs -keyalg rsa -alias vdt_alias -storepass server_ks_pwd -keypass server_pwd
```
The keytool utility prompts to enter details about the certificate. Provide answers on the command-line when prompted.

2.2.5.2 Building Veridata Server and Veridata Agent Keystores
To build the Veridata Agent keystore, run the following keytool command:
```shell
keytool -genkey -alias agent.server.keys -keyalg RSA -keystore agent.server.keystore -storepass ks_password -keypass keypwd
```
To export the Veridata Agent certificate to a file, run the following keytool command:
```shell
keytool -export -alias agent.server.keys -keystore agent.server.keystore -storepass ks_password -file agent.server.cer
```
To build the Veridata Web Server keystore, run the following keytool command:
```shell
```
To export the Veridata Server certificate to a file, run the following keytool command:
```shell
```

2.2.5.3 Importing Certificates to the Veridata Server and Agent Truststores
To import the Veridata Server certificate to the Agent truststore, run the following keytool command:
```shell
keytool -import -v -keystore agent.server.truststore -storepass ks_password -file vdt.web.client.cer
```
To import the Veridata Agent certificate the Veridata Web Server truststore, run the following keytool command:
```shell
keytool -import -v -keystore vdt.web.client.truststore -storepass ks_password -file agent.server.cer
```

2.2.5.4 Examples

Example 1 Create a Veridata Agent ID Keystore
```shell
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 changeit -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 Veridata 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]: VERIDATA WEBLOGIC 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 vdtServerID.cer file.

Example 3 Create a Trust Stores for Veridata 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.server.id

Owner: CN=VERIDATA WEBLOGIC SERVER, OU=NA, O=COMPANY A, L=USA, ST=USA, C=US
Issuer: CN=VERIDATA WEBLOGIC SERVER, OU=NA, O=COMPANY A, L=USA, ST=USA, C=US
Serial number: 2aded02f

2-6 Administering Oracle GoldenGate Veridata
Certificate fingerprints:
   SHA256:
   B: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.agent.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:
   SHA256:
   8:5B: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]
### 2.2.6 Using OPSS Keystore Service to Manage Veridata Keystores

The Oracle Platform Security Services (OPSS) keystore services is used as a repository for storing identity and trust keystores for Veridata Server. The Veridata Agent keystores can also be managed using the OPSS keystore service. For more information, see “Managing Keys and Certificates with the Keystore Service” in *Securing Applications with Oracle Platform Security Services*.

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

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name of the Application Stripe created by WebLogic Server.</td>
<td>VeridataSec</td>
</tr>
<tr>
<td>Name of the identity keystore under the application stripe VeridataSec</td>
<td>VeridataWebIdentityStore</td>
</tr>
<tr>
<td>Name of the trust keystore under the application stripe VeridataSec</td>
<td>VeridataWebTrustStore</td>
</tr>
</tbody>
</table>

To configure two-way SSL by using the OPSS keystore service:

1. For each Veridata Agent, create an identity and trust keystore.
2. Update the `VeridataWebIdentityStore` with the identity certificate of the Veridata Web Server.
3. Update the `VeridataWebTrustStore` with all Veridata Agent certificates.
4. Update each Veridata Agent truststore with the identity certificate of the Veridata Web 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 OPSS keystore service:

1. For each Veridata Agent, create an identity keystore.
2. Update the `VeridataWebIdentityStore` with the identity certificate of the Veridata Web Server.
3. Export the Agent keystore and truststore as JKS files and note 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.

### 2.2.7 Modifying the Veridata Agent Wallet

Before you start the Veridata agent in SSL mode, you must update the Veridata Agent Wallet with the identity and trust keystore passwords. Otherwise, the Veridata Agent fails to start.
To update the wallet, run the `configure_agent_ssl` script that is available in the agent home:

```
AGENT_HOME\configure_agent_ssl.sh AgentID
```

where `AgentID` is the name of the agent properties file, without the `.properties` extension. The default value for `AgentID` is `agent`.

When prompted, enter the entry or unlock password for the identity and trust keystores for the agent.

### 2.3 Securing the Oracle GoldenGate Veridata Files

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

#### 2.3.1 Controlling Access to the Installation Directories

Standard operating system permissions apply to the programs, files, and directories within the Oracle GoldenGate Veridata Server and Web User Interface, and Oracle GoldenGate Veridata Agent installation directories. You should adjust the permissions for these objects based on your business security rules.

#### 2.3.2 Securing files that Contain Business Data

Oracle GoldenGate Veridata Server creates data files that will contain sensitive application data. By default, these files reside in the `DOMAIN_HOME/veridata/reports`. All of the sub-directories within that directory contain files that may reflect business data.

The types of files that contain sensitive data are:

- The comparison report (`rpt` sub-directory)
- The out-of-sync report (`oosxml` and `oos` sub-directories)

These files inherit the same file permissions as those of the user that runs the Oracle GoldenGate Veridata Server installation program. Do not change the permissions, or Oracle GoldenGate Veridata may be unable to maintain them. These files should be kept just as secure as you would keep your business data. Users of Oracle GoldenGate Veridata Web do not require access to these files because they see the same information through the client interface.

The contents of all report files can be optionally encrypted. For more information, see Section 2.6, "Encrypting Report Files".

### 2.4 Securing Access to Oracle GoldenGate Veridata by Defining User Roles

You can assign security roles to the users of Oracle GoldenGate Veridata to control their access to the functions that are performed by the software, some of which expose selected data values from the database. Table 2–4 describes the Veridata user roles.
Table 2–4  Veridata User Roles

<table>
<thead>
<tr>
<th>Veridata</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>veridataAdministrator</td>
<td>Type-A</td>
<td>The administrator 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.</td>
</tr>
<tr>
<td>veridataPowerUser</td>
<td>Type-A</td>
<td>The power user 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.</td>
</tr>
<tr>
<td>veridataReportViewer</td>
<td>Type-B</td>
<td>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.</td>
</tr>
<tr>
<td>veridataDetailReportViewer</td>
<td>Type-B</td>
<td>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, and view comparison reports and out-of-sync report information through the Oracle GoldenGate Veridata Web User Interface or at the file level.</td>
</tr>
<tr>
<td>veridataRepairOperator</td>
<td>Additional</td>
<td>The RepairOperator role can use the Repair feature in Veridata.</td>
</tr>
<tr>
<td>veridataCommandLineUser</td>
<td>Additional</td>
<td>The commandLineUser role provides access to the Veridata command line tools, vericom and the Veridata import and export utilities.</td>
</tr>
</tbody>
</table>

These roles are categorized into various types 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 Veridata functions.

Security is controlled through the Oracle WebLogic Server Administration Console. From this interface, 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.

**To open Oracle WebLogic Server Administration Console**

1. Connect to the Oracle WebLogic Server Administration Console from a browser by typing the following address:

   http://weblogic_admin_server_hostname:admin_server_port/console

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

2. Log on to the Administration Console as an Oracle GoldenGate Veridata administrator user. A default administrator user was created during the creation of Oracle GoldenGate Veridata domain.

**To create or edit a user**

1. In the left pane of the Administration Console, select **Security Realms**.

2. On the Summary of Security Realms page select the name of the Veridata security realm.

3. On the Settings for Veridata Security realm page select **Users and Groups > Users**. The User table displays the names of all users defined in the Authentication provider. Click New to create a new user or select an existing user to edit settings. Enter the following properties for a new user:
   - **Name**: Specify a name for the user.
   - **Provider**: Select the Authentication provider for the user.
   - **Password**: Enter a password for the user.
   - **Description**: (Optional) Specify a description for this user.

4. To assign a role to the user, go to the Settings for **user_name** page and click Groups. Select appropriate roles for the user. All roles for a Veridata user are described in Table 2–4, "Veridata User Roles".

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

   ![Settings for VeridataAdmin](image)

For example, a Veridata power user, **PowerUser**, can be given privileges as shown in the figure below.
Changing Database Schema Passwords

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

Change a database schema password:

1. Stop your Veridata Server.
   
   ```bash
   DOMAIN_HOME/veridata/veridata/bin/veridataServer.sh stop
   ```

2. Stop your Oracle WebLogic Server.
   
   ```bash
   DOMAIN_HOME/veridata/bin/stopWebLogic.sh
   ```

3. Start the Oracle WebLogic Scripting Tool, `wlst.sh`. For example:
   
   ```bash
   /home/oracle/Oracle/Middleware/Oracle_Home/oracle_common/common/bin/wlst.sh
   ```

4. Modify the database schema password as in this example:
   
   ```bash
   modifyBootStrapCredential(jpsConfigFile='/home/oracle/wls_domains/veridata/config/fmwconfig/jps-config.xml',username='OGG_OPSS',password='welcome123')
   ```

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

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

6. Log into the metadata repository database and change the schema passwords and unlock the schema as in the following example that unlocks the OGG_OPSS schema:
   
   ```sql
   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 Configuration Wizard, config.sh. For example:
   
   /home/oracle/Oracle/Middleware/Oracle_Home/oracle_common/common/bin/config.sh

8. Select Update an Existing Domain then click Next.

9. Select JDBC Component Schema then enter the new database schema passwords.

10. Click Next until you reach the Configuration Summary screen, and then click Update.

11. Click Next then Finish to save your schema password changes.

   
   DOMAIN_HOME/veridata/bin/startWebLogic.sh

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

2.6 Encrypting Report Files

Oracle GoldenGate Veridata provides you with an option to encrypt the comparison report files (.rpt, .oos, .oosxml). The following sections explain the report encryption in Veridata:

- Section 2.6.1, "Enabling Report Encryption"
- Section 2.6.2, "Using the reportutil Tool to view Reports"

2.6.1 Enabling Report Encryption

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

- server.encryption
- server.encryption.bits

To enable encryption of Veridata report files, set server.encryption to true.

Encryption of Veridata report files use 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 server.encryption.bits parameter in veridata.cfg. Note that any encryption strength greater than 128 requires you to use a JRE that has Unlimited Strength Cryptography Extension installed.

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

Encrypted Veridata 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)
2.6.2 Using the reportutil Tool to view Reports

When Veridata report encryption is enabled, all Veridata report files will be encrypted using an encryption key which is initially a large random value. The encryption key can be changed if required.

The encrypted files must be decrypted before you read it. The Veridata Web User Interface automatically decrypts files before displaying them. Alternatively, use the reportutil.sh/.bat utility to display the encrypted contents. This utility is located in the VERIDATA_DOMAIN_HOME\veridata\bin directory. Run the tool as follows:

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

Where `wlport` is the WebLogic Server port number (the default port is 8830) and `wluser` represents the WebLogic 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. Or else the command prints the decrypted file to the standard output.

Note that the Veridata user running the `reportutil` tool must be in the appropriate user group to perform the operations:

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

For more information about the user roles, see Section 2.4, "Securing Access to Oracle GoldenGate Veridata by Defining User Roles".
3

Running the Oracle GoldenGate Veridata Programs

This chapter explains how to run the Oracle GoldenGate Veridata programs, such as the agents and Java components.

This chapter includes the following sections:
- Starting and Stopping the C-agent and Manager
- Starting and Stopping the Java-Based Components
- Reloading Log Information
- Connecting to Oracle GoldenGate Veridata Web Interface

3.1 Starting and Stopping the C-agent and Manager

The C-agent starts automatically at the request of Oracle GoldenGate Veridata Server when initiating comparisons. However, for Oracle GoldenGate Veridata 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 an automatic process, you can stop the Manager process that controls the agent. Stopping Manager prevents Oracle GoldenGate Veridata 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 Oracle GoldenGate Veridata Agent installation location, run GGSCI.
2. In GGSCI, issue the appropriate command as follows to stop or start the Manager.

   START MANAGER

   Or...

   STOP MANAGER

3.2 Starting and Stopping the Java-Based Components

The Oracle GoldenGate Veridata Server and Oracle GoldenGate Veridata Web components are Java-based programs. The Oracle GoldenGate Veridata 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.

3.2.1 Controlling the Java-Based Components from the Command Line

To control the agent component, change directories to its installation directory and issue the appropriate command as follows:

<table>
<thead>
<tr>
<th>UNIX and Linux</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>agent.sh {start</td>
<td>run}</td>
</tr>
<tr>
<td>Or...</td>
<td>Or...</td>
</tr>
<tr>
<td>agent.sh stop</td>
<td>agent.bat stop</td>
</tr>
</tbody>
</table>

Where:

- **run** starts the agent in the same command window from which 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.

3.3 Reloading Log Information

You can reload logging information from the `AGENT_ORACLE_HOME/config/odl.xml` configuration file to a running agent by using the `reloadLog` option. The changes in the `odl.xml` file are put into effect on the agent. The agent must be running for this command to work.

Use the following command:

<table>
<thead>
<tr>
<th>UNIX and Linux</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>agent.sh reloadLog</td>
<td>agent.bat reloadLog</td>
</tr>
</tbody>
</table>

3.4 Connecting to Oracle GoldenGate Veridata Web Interface

To connect to the Veridata Web User Interface, open a web browser and type the following address:

http://hostname:port/veridata

Where:
hostname is the name of the system where Oracle GoldenGate Veridata Server is installed and port is the port number where it is running (default is 8830). Use localhost as the host name if connecting on the system that is local to the server installation.

Examples:

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

The Oracle GoldenGate Veridata Web login page is displayed upon successful connection. Log in with your user name and password. For full instructions on using Oracle GoldenGate Veridata Web Interface, see the Online Help.
This chapter explains how to use the `vericom` command line interface to run comparisons.

This chapter includes the following sections:

- Overview of the `vericom` Tool
- Running `vericom`
- `vericom` exit statuses
- `vericom` Output Examples

## 4.1 Overview of the `vericom` Tool

You can use the `vericom` tool of Oracle GoldenGate Veridata to execute certain comparison tasks from the command shell of the operating system. The `vericom` tool runs the Oracle GoldenGate Veridata Command Line Interface and enables you to handle these activities with automated programs.

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 interface

Comparisons also can be run from the Oracle GoldenGate Veridata Web interface. This interface provides greater control in configuring the objects to be compared and for controlling runtime parameter settings.
4.2 Running Vericom

The `vericom` program can be run by anyone who has the correct operating system permissions to run it.

1. On the system where the Oracle GoldenGate Veridata is installed, run the command shell of the operating system.
2. Navigate to the `VERIDATA_DOMAIN_HOME/veridata/bin` directory.
3. Use the following syntax to run the `vericom` program.

**Syntax**

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

**Required Parameters**

One of the following are required; otherwise an error is returned. Enter only one option.

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

The `-wluser` specifies the WebLogic Server user name to connect to the WebLogic Server. This WebLogic 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 Veridata Import and Export utilities.

See Section 2.4, "Securing Access to Oracle GoldenGate Veridata by Defining User Roles".

If `-version`, `-v`, `-help`, or `-helprun` are specified, they take precedence over any other flag 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 ]  
[ -rTs 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 ]

Table 4–1 Vericom Runtime Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-wluser</td>
<td>Specifies the WebLogic Server user name that authenticates and connects to the server.</td>
</tr>
<tr>
<td>-wlport</td>
<td>Specifies the WebLogic Server port number.</td>
</tr>
<tr>
<td>-help</td>
<td>Displays the vericom syntax components and their descriptions.</td>
</tr>
<tr>
<td>-helprun</td>
<td>Displays run-related syntax components and their descriptions.</td>
</tr>
<tr>
<td>(-version</td>
<td>-v)</td>
</tr>
<tr>
<td>(-job</td>
<td>-j) job</td>
</tr>
<tr>
<td>-g group</td>
<td>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.</td>
</tr>
<tr>
<td>-c compare_pair</td>
<td>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.</td>
</tr>
<tr>
<td>-mw</td>
<td>Directs vericom not to wait for the job to finish before returning the prompt. Instead, vericom returns immediately after starting a job.</td>
</tr>
<tr>
<td>-repair</td>
<td>-norepair</td>
</tr>
<tr>
<td>-rP profile</td>
<td>Overrides the 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.</td>
</tr>
<tr>
<td>-rR</td>
<td>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.</td>
</tr>
<tr>
<td>-rO</td>
<td>A run override option. Generates an OOSXML file 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.</td>
</tr>
</tbody>
</table>

■ If -g and -c are used, -j must also be used.

■ If -rP is used, -j must be used.

■ Do not use -rR and -rO in the same run.
Table 4–1 (Cont.) Vericom Runtime Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-rN threads</td>
<td>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.</td>
</tr>
<tr>
<td>-rD seconds</td>
<td>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 the -rR option is used.</td>
</tr>
<tr>
<td>-rC</td>
<td>Controls whether or not the confirmation step (confirm OOS) is performed in the job.</td>
</tr>
<tr>
<td>-rC</td>
<td>-rC skips the confirmation step. You can skip the confirmation step if activity on the source tables is stopped or if replication is not continuously updating the target table(s).</td>
</tr>
<tr>
<td>+rC</td>
<td>+rC includes the confirmation step.</td>
</tr>
<tr>
<td>These options override the default job profile and are mutually exclusive. They have no effect unless -j is used.</td>
<td></td>
</tr>
<tr>
<td>-rOb</td>
<td>-rOx</td>
</tr>
<tr>
<td>-rOb generates binary format that is compatible with the Oracle GoldenGate Veridata Web browser.</td>
<td></td>
</tr>
<tr>
<td>-rOx generates output in XML.</td>
<td></td>
</tr>
<tr>
<td>-rO2 generates both binary and XML output.</td>
<td></td>
</tr>
<tr>
<td>-rO0 suppresses out-of-sync output.</td>
<td></td>
</tr>
<tr>
<td>These options override the default job profile and are mutually exclusive. They have no effect unless -rR is used.</td>
<td></td>
</tr>
<tr>
<td>-rOs records</td>
<td>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 -rR is used.</td>
</tr>
<tr>
<td>-rTi</td>
<td>Turns on tracing of Oracle GoldenGate Veridata Agent for the initial comparison step. Do not use without the guidance of an Oracle support analyst.</td>
</tr>
<tr>
<td>-rTc</td>
<td>Turns on tracing of Oracle GoldenGate Veridata Agent for the confirmation step. Do not use without the guidance of an Oracle support analyst.</td>
</tr>
<tr>
<td>-rTs trace_number</td>
<td>Turns on tracing for Oracle GoldenGate Veridata Server. trace_number is a bitmask of server execution trace flags. The higher the level, the more detailed the trace data. Do not use without the guidance of an Oracle support analyst.</td>
</tr>
</tbody>
</table>
Running Vericom

Running Comparisons from the Command Line

4-5

-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 Veridata Server. There will be an error 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), 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.

-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.

Table 4–1 (Cont.) Vericom Runtime Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-pS source_partition_name</td>
<td>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.</td>
</tr>
<tr>
<td>-pSq source_sql_predicate</td>
<td>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 Veridata Server. There will be an error if the specified partition does not exist.</td>
</tr>
<tr>
<td>-pSA1 source_ascii_start_key</td>
<td>Specifies an ASCII key as the starting key value of a partition that overrides an existing source partition for an Enscribe file.</td>
</tr>
<tr>
<td>-pSA2 source_ascii_end_key</td>
<td>Specifies an ASCII key as the ending key value of a partition that overrides an existing source partition for an Enscribe file.</td>
</tr>
<tr>
<td>-pSH1 source_hex_start_key</td>
<td>Specifies a hexadecimal key as the starting key value of a partition that overrides an existing source partition for an Enscribe file.</td>
</tr>
<tr>
<td>-pSH2 source_hex_end_key</td>
<td>Specifies a hexadecimal key as the ending key value of a partition that overrides an existing source partition for an Enscribe file.</td>
</tr>
<tr>
<td>-pT target_partition_name</td>
<td>These options specify target partitions and have the same rules as the corresponding options that specify source partitions.</td>
</tr>
<tr>
<td>-pTq target_sql_predicate</td>
<td>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.</td>
</tr>
</tbody>
</table>
4.3 Vericom exit statuses

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

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The command executed successfully. If a job was run, it finished with all rows in-sync. If -nw was specified, the exit status is 0 if the job started successfully.</td>
</tr>
<tr>
<td>1</td>
<td>Invalid vericom syntax was used. For example, the following are invalid: vericom.sh -helptun (Typographical error.) vericom.sh -j -g group1 (The name of the job is missing.)</td>
</tr>
<tr>
<td>3</td>
<td>Provides more granularity for input errors that involve flags that run comparisons. For example, the following mistakes will 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.</td>
</tr>
<tr>
<td>4</td>
<td>The job ran successfully, but there were rows that had a comparison status of something other than in-sync.</td>
</tr>
<tr>
<td>5</td>
<td>There was a communication error with Oracle GoldenGate Veridata Server.</td>
</tr>
</tbody>
</table>

4.4 Vericom Output Examples

To view the results of a comparison that is run with vericom, you can use Oracle GoldenGate Veridata Web to view the comparison report (see Section 1.4, "Viewing Comparison Results"), and you can view the output that is returned by vericom to the terminal. If a run finishes successfully, statistics for the job are displayed.
Example 1
The following example shows a run on a Windows system of TestJob without specifying -w. The process exits with status 0, and finished job statistics are not displayed.

VERIDATA_DOMAIN_HOME\veridata\bin\vericom.bat -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
The following example shows a run of 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 -j TestJob -w
Connecting to: localhost:9177
Run ID: (2257, 0, 0)
Job Start Time: 2008-03-21 22:48:05
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
The following example shows a run of compare pair 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 -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 of the previously executed command is in the special variable `'$?` if you are using SH or KSH shells. If you are using the CSH shell, the exit status of the previously executed command is in the special variable `'$status'`. 
This chapter describes how to use 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**
- **Running the Import and Export Utilities**
- **Configuration File Element Reference**

### 5.1 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

### 5.1.1 Supported Configurations

Oracle GoldenGate Veridata import and export utilities support configuring:

- Database connections
5.2 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`.

5.2.1 Using the Export Utility

The syntax for running the export utility is:

```
veridata_export[.sh | .bat] -export filePath -wluser commandlineUsername
[-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.
Running the Import and Export Utilities

5.2.2 Using the Import Utility

The syntax for running the import utility is:

```
veridata_import[.sh | .bat] [-wlport portNo] -wluser commandlineUsername [-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.

5.2.3 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.
5.3 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.

```xml
<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>

Appendix B provides a sample configuration file.

Appendix C provides a description of the profile parameters that you can use to configure your profiles.
The root element is **configuration**.

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

<table>
<thead>
<tr>
<th>Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connection</td>
<td>One or more Veridata database connection definitions.</td>
</tr>
<tr>
<td>group</td>
<td>One or more Veridata comparison group definitions.</td>
</tr>
<tr>
<td>job</td>
<td>One or more comparison job definitions.</td>
</tr>
<tr>
<td>profile</td>
<td>One or more profile definitions.</td>
</tr>
</tbody>
</table>

The following attributes describe the **configuration** element:

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source-name</td>
<td>A regular expression that defines a set of source column names. This value is required.</td>
</tr>
<tr>
<td>target-name</td>
<td>A regular expression that defines a set of target column names. It can include references to groups captured by the source-name expression.</td>
</tr>
</tbody>
</table>
| 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". |
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**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>colfiltercol</code></td>
<td>Specifies a set of columns to be included or excluded.</td>
</tr>
</tbody>
</table>

The following attribute describes the `colfilter` element:

**Table 5-5 colfilter Attribute**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Specifies whether to include the columns or exclude them. The options are include or exclude; the default is include. This is a required attribute.</td>
</tr>
</tbody>
</table>

**Example**

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

```xml
<filter type='include' table='TABLE_NAME'>
  <colfilter type='exclude'>
    <colfiltercol name='COL3' />
    <colfiltercol name='COL5' />
  </colfilter>
</filter>
```
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>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>A regular expression that defines a set of source column names. This is a required attribute.</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Table 5–7 compare-pair Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>enscribe-info</td>
</tr>
<tr>
<td>sql-partition</td>
</tr>
<tr>
<td>enscribe-key</td>
</tr>
<tr>
<td>key-column</td>
</tr>
<tr>
<td>column</td>
</tr>
<tr>
<td>excluded-column</td>
</tr>
<tr>
<td>delta-config</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Table 5–8 compare-pair Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attribute</strong></td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>source-table</td>
</tr>
<tr>
<td>target-table</td>
</tr>
<tr>
<td>source-schema</td>
</tr>
<tr>
<td>target-schema</td>
</tr>
<tr>
<td>source-catalog</td>
</tr>
<tr>
<td>target-catalog</td>
</tr>
</tbody>
</table>
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:
Using the Veridata Import and Export Utilities

- \( P(.*)) \) - 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, \texttt{S_TABLE} and \texttt{T_TABLE}, where \texttt{S_TABLE} is a table in schema "test" and \texttt{T_TABLE} is table in schema "other". It also excludes all non-key columns in the generated compare pairs.

```xml
<configuration>
  <connection name="source" host="somehost"
    .... use-ssl="true">
    <description>
      <![CDATA[
        Group SQL Scripting Source Connection
      ]]>}
    </description>
  </connection>
...
...
</configuration>
```
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>
<thead>
<tr>
<th>Table 5–9 connection Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>description</td>
</tr>
<tr>
<td>conn-properties</td>
</tr>
</tbody>
</table>

The following attributes describe the connection element:

<table>
<thead>
<tr>
<th>Table 5–10 connection Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attribute</strong></td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>host</td>
</tr>
<tr>
<td>port</td>
</tr>
<tr>
<td>user</td>
</tr>
<tr>
<td>password</td>
</tr>
<tr>
<td>repairUser</td>
</tr>
<tr>
<td>repairPassword</td>
</tr>
<tr>
<td>agent-timeout</td>
</tr>
<tr>
<td>truncate-spaces</td>
</tr>
<tr>
<td>fetch-size</td>
</tr>
<tr>
<td>use-ssl</td>
</tr>
</tbody>
</table>

Example

The following example identifies the connection named source.

```xml
<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>
</configuration>
```
</connection>
.
.
</configuration>
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>
<thead>
<tr>
<th>Table 5–11 conn-properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>datatype-name</td>
</tr>
<tr>
<td>format</td>
</tr>
<tr>
<td>precision</td>
</tr>
<tr>
<td>scale</td>
</tr>
<tr>
<td>timezone</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source-config</td>
<td>Provides source side configuration for delta processing.</td>
</tr>
<tr>
<td>target-config</td>
<td>Provides target side configuration for delta processing.</td>
</tr>
<tr>
<td>query</td>
<td>Specifies the query for delta processing.</td>
</tr>
</tbody>
</table>

**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.

```xml
<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 ]]]>
        </source-config>
      </delta-config>
    </compare-pair>
  </group>

</configuration>
```
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>
<thead>
<tr>
<th>Table 5–13  enscribe-info Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
</tr>
<tr>
<td>expantddl</td>
</tr>
</tbody>
</table>

The following attributes describe the enscribe-info element:

<table>
<thead>
<tr>
<th>Table 5–14  enscribe-info Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
</tr>
<tr>
<td>side</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>dictionary</td>
</tr>
<tr>
<td>record</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>A name that identifies the key. This is a required attribute.</td>
</tr>
<tr>
<td>start-key</td>
<td>The key that is to be used to begin reading the Enscribe file. This is a required entry.</td>
</tr>
<tr>
<td>end-key</td>
<td>The key of the last Enscribe record that should be read. This is a required entry.</td>
</tr>
<tr>
<td>format</td>
<td>Specifies the format of the Enscribe key. The options are: &quot;ascii&quot; - The format of the key is ASCII. This is the default. &quot;hexadecimal&quot; - The format of the key is hexadecimal.</td>
</tr>
<tr>
<td>side</td>
<td>Indicates whether the partition should be applied at the source database, the target database, or both databases.</td>
</tr>
<tr>
<td>default</td>
<td>Indicates whether this is the default partition. This is equivalent to the &quot;use at run time&quot; indicator on the UI. The default is both.</td>
</tr>
</tbody>
</table>

Examples

```xml
<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'/>```
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**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>A regular expression that defines a set of source column names. This is a required attribute.</td>
</tr>
</tbody>
</table>
The `expandddl` element describes the rules used when applying the DDL.

The following attributes describe the `expandddl` element:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expandGroupArrays</td>
<td>Whether or not to expand group arrays. The options are:</td>
</tr>
<tr>
<td></td>
<td>&quot;true&quot; to expand the array. This is the default.</td>
</tr>
<tr>
<td></td>
<td>&quot;false&quot; not to expand the array.</td>
</tr>
<tr>
<td>redefined-columns</td>
<td>Whether or not to include redefined columns. The options are:</td>
</tr>
<tr>
<td></td>
<td>&quot;include&quot; - Includes redefined columns</td>
</tr>
<tr>
<td></td>
<td>&quot;omit&quot; - Leaves out redefined columns. This is the default.</td>
</tr>
<tr>
<td>resolvedups</td>
<td>Specifies how to resolve duplicates that result when the array is expanded.</td>
</tr>
<tr>
<td></td>
<td>The options are:</td>
</tr>
<tr>
<td></td>
<td>&quot;appendIndex&quot; - Adds a unique numeric index to the end of the duplicate.</td>
</tr>
<tr>
<td></td>
<td>&quot;appendAlphaIndex&quot; - Adds an alpha character index to the end of the</td>
</tr>
<tr>
<td></td>
<td>duplicate.</td>
</tr>
<tr>
<td></td>
<td>&quot;prependGroup&quot; - Prefixes the name of the array group to the duplicate</td>
</tr>
<tr>
<td>ddl-separator</td>
<td>The character separator for defining array output into columns.</td>
</tr>
<tr>
<td></td>
<td>An example is the dash used in FIELDX-3, which is the third occurrence of</td>
</tr>
<tr>
<td></td>
<td>FIELDX in the array. The options are:</td>
</tr>
<tr>
<td></td>
<td>&quot;none&quot; - There is no separator. This is the default.</td>
</tr>
<tr>
<td></td>
<td>&quot;dash&quot; - Use a dash (-) as the separator.</td>
</tr>
<tr>
<td></td>
<td>&quot;bracket&quot; - Use brackets [ ] as the separator.</td>
</tr>
<tr>
<td></td>
<td>&quot;underscore&quot; - Use underscore (__) as the separator.</td>
</tr>
<tr>
<td></td>
<td>&quot;double-underscore&quot; - Use double underscore (__) as the separator</td>
</tr>
<tr>
<td>zero-fill-length</td>
<td>Prepends zeros to adjust the number of the occurrence. The value is the</td>
</tr>
<tr>
<td></td>
<td>number of digits enclosed in quotation marks. &quot;0&quot; is the default.</td>
</tr>
<tr>
<td>fix-long-names</td>
<td>Whether to fix the names that result from resolving duplicates if they</td>
</tr>
<tr>
<td></td>
<td>exceed the max-col-name-length. The options are:</td>
</tr>
<tr>
<td></td>
<td>&quot;true&quot; - Fix the names that exceed the maximum. This is the default.</td>
</tr>
<tr>
<td></td>
<td>&quot;false&quot; - Do not change the names that exceed the maximum.</td>
</tr>
<tr>
<td>max-col-name-length</td>
<td>The maximum length allowed for a column name. The entry is a number within</td>
</tr>
<tr>
<td></td>
<td>quotation marks. The default is ‘120’.</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Specifies either to include or exclude schemas and tables. Valid values are include or exclude.</td>
</tr>
<tr>
<td>catalog</td>
<td>Specifies the default catalog name.</td>
</tr>
<tr>
<td>schema</td>
<td>Specifies the schema name.</td>
</tr>
<tr>
<td>table</td>
<td>Specifies the table name.</td>
</tr>
<tr>
<td>file-pattern</td>
<td>For NonStop platforms only, specifies the file pattern.</td>
</tr>
</tbody>
</table>

**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>
```
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>
<thead>
<tr>
<th>Table 5–19  Group Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>description</td>
</tr>
<tr>
<td>filter</td>
</tr>
<tr>
<td>sql-partition</td>
</tr>
<tr>
<td>enscribe-key</td>
</tr>
<tr>
<td>compare-pair</td>
</tr>
</tbody>
</table>

The following attributes describe the `group` element:

<table>
<thead>
<tr>
<th>Table 5–20  Group Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attribute</strong></td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>source-conn</td>
</tr>
<tr>
<td>target-conn</td>
</tr>
<tr>
<td>source-schema</td>
</tr>
<tr>
<td>target-schema</td>
</tr>
<tr>
<td>source-catalog</td>
</tr>
<tr>
<td>target-catalog</td>
</tr>
</tbody>
</table>
Table 5–20  (Cont.) Group Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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**

```xml
<group name="weekly-tables" source-conn="source" target-conn="target">
  <description>
    
  </description>
  <sql-partition>
    
  </sql-partition>
  <compare-pair>
    
  </compare-pair>
</group>
```
The **job** element defines an Oracle GoldenGate Veridata comparison job.

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

<table>
<thead>
<tr>
<th>Table 5–21</th>
<th><strong>job Elements</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Description</td>
</tr>
<tr>
<td>description</td>
<td>Provides a description of the job.</td>
</tr>
<tr>
<td>group</td>
<td>The name of the group associated with the job. This can be a new group or a previously defined group.</td>
</tr>
</tbody>
</table>

The following attributes describe the **job** element:

<table>
<thead>
<tr>
<th>Table 5–22</th>
<th><strong>job Attributes</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>name</td>
<td>A name that identifies the job. This is a <em>required</em> attribute.</td>
</tr>
<tr>
<td>source-conn</td>
<td>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 <strong>job</strong> <em>source-conn</em> is used to override the source connection specified for the groups included in the job.</td>
</tr>
<tr>
<td>target-conn</td>
<td>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.</td>
</tr>
<tr>
<td>profile</td>
<td>The default profile to use when running the job.</td>
</tr>
</tbody>
</table>

**Example**

```xml
<job name="all-groups" profile="server-sort">
  <group name="all-tables"/>
  <group name="selected-tables"/>
</job>
```
The `profile` element defines the connection properties of a comparison job connection.

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

<table>
<thead>
<tr>
<th>Table 5–23</th>
<th>profile Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Description</td>
</tr>
<tr>
<td>description</td>
<td>Provides a description of the profile.</td>
</tr>
<tr>
<td>profile-general</td>
<td>Defines the profile parameters that control the output options.</td>
</tr>
<tr>
<td>sorting-method</td>
<td>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.</td>
</tr>
<tr>
<td>initial-compare</td>
<td>Defines the profile parameters that control the parameters for the job that performs the initial compare step</td>
</tr>
<tr>
<td>confirm-out-of-sync</td>
<td>Specifies the profile parameters that control the parameters for the job that performs the confirmation step</td>
</tr>
<tr>
<td>repair</td>
<td>Specifies the profile parameters that control the parameters for the repair job.</td>
</tr>
</tbody>
</table>

The following attributes describe the `profile` element:

<table>
<thead>
<tr>
<th>Table 5–24</th>
<th>profile Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>name</td>
<td>A name that identifies the profile. This is a required attribute.</td>
</tr>
</tbody>
</table>

**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)

```xml
<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>
```
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**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source-name</td>
<td>A regular expression that defines a set of source column names. This value is required.</td>
</tr>
<tr>
<td>target-name</td>
<td>A regular expression that defines a set of target column names. It can include references to groups captured by the <code>source-name</code> expression.</td>
</tr>
<tr>
<td>format</td>
<td>Specifies a format to override the comparison format that would normally be used.</td>
</tr>
<tr>
<td>scale</td>
<td>Specifies a scale to override the default scale for the comparison.</td>
</tr>
<tr>
<td>precision</td>
<td>Specifies a precision to override the default precision used for the comparison.</td>
</tr>
<tr>
<td>timezone</td>
<td>Specifies a time zone to override the default time zone of the comparison.</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Table 5–26  profile-general Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
</tr>
<tr>
<td>param</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>param</td>
<td>Defines the parameter to change for the profile.</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Table 5–28</th>
<th>initial-compare Element</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>param</td>
<td>Defines the parameter to change for the profile.</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Table 5–29 confirm-out-of-sync Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>param</td>
</tr>
</tbody>
</table>
The `param` element defines the parameters that are used for configuring profile options.

The following attributes describe the `repair` element:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the parameter. This is a required attribute.</td>
</tr>
<tr>
<td>value</td>
<td>The value of the parameter</td>
</tr>
</tbody>
</table>
The `repair` element provides parameters for the repair process.

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

<table>
<thead>
<tr>
<th>Table 5–31  repair Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>param</td>
<td>Defines the parameters that are used to configure the profile options.</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>A name that identifies the partition. This is a required attribute.</td>
</tr>
<tr>
<td>side</td>
<td>Indicates whether the partition should be applied at the source database, the target database, or both databases. The default is &quot;both&quot;.</td>
</tr>
<tr>
<td>default</td>
<td>Indicates whether this is the default partition. This is equivalent to the &quot;use at run time&quot; indicator on the UI. The default is &quot;false&quot;.</td>
</tr>
</tbody>
</table>

Example

```xml
<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>
```
This chapter explains how to use the Veridata GoldenGate Parameter Processing (VGPP) command line tool to use Oracle GoldenGate parameter files.

This chapter includes the following sections:

- Section 6.1, "Overview of the Command Line Interface"
- Section 6.2, "Running VGPP"
- Section 6.3, "Parameter Handling"
- Section 6.4, "Map and Table Statement Handling"

### 6.1 Overview of the Command Line Interface

An Oracle GoldenGate parameter file contains all of the information required to extract or apply replicated data. The data propagated through Oracle GoldenGate replication is controlled by settings in the Extract and/or Replicat parameter files. Typically, the Extract parameter file specifies the tables to be replicated and Replicat parameter file controls the mapping of columns, as well as further restricting the tables. The VGPP command line utility accepts one or two Oracle GoldenGate parameter files as input. One of these parameter files must be a Replicat parameter file while the other optional file must be an Extract parameter file.

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

Oracle GoldenGate Veridata only support single column mapping. Thus, the column mapping assumes a one to one mapping between source columns and target columns.

You can:

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

The VGPP program can be run by anyone who has the correct operating system permissions to run it.

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. Use the following syntax to run the VGPP program.

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 specifies the WebLogic Server user name to connect to the WebLogic Server. This WebLogic 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 Veridata import and export utilities.

Optional Parameter

This is the optional parameter:

[extract_param_filename]

Table 6-1 VGPP Runtime Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ -noscripting</td>
<td>-create</td>
</tr>
<tr>
<td>-wluser</td>
<td>Specifies the WebLogic Server user name that authenticates and connects to the server.</td>
</tr>
<tr>
<td>-wlport</td>
<td>Specifies the WebLogic Server port number. The default listening port is 8830.</td>
</tr>
<tr>
<td>-p</td>
<td>Specifies a properties file containing additional information required for the Veridata configuration.</td>
</tr>
</tbody>
</table>
6.2.1 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

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>source.connection.name</td>
<td>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.</td>
</tr>
<tr>
<td>source.connection.port</td>
<td>The port for the source agent. This is required when the connection does not already exist.</td>
</tr>
<tr>
<td>source.connection.host</td>
<td>The host name where the source agent is running. This is required when the connection does not exist.</td>
</tr>
<tr>
<td>source.connection.user</td>
<td>This defaults to the user information in the extract parameter file. This is required when the connection does not exist.</td>
</tr>
<tr>
<td>source.connection.password</td>
<td>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.</td>
</tr>
<tr>
<td>source.catalog</td>
<td>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 SOURCENAME parameter or the first part of a three-part name in a TABLE statement.</td>
</tr>
<tr>
<td>extract.useansiquotes</td>
<td>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</td>
</tr>
<tr>
<td>extract.charset</td>
<td>The character set for the extract parameter file. This overrides any charset specified in the extract parameter file.</td>
</tr>
</tbody>
</table>
6.3 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**

<table>
<thead>
<tr>
<th>GoldenGate Parameters</th>
<th>Veridata Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATALOGEXCLUDE</td>
<td>This parameter is ignored. Veridata only processes items from a single catalog.</td>
</tr>
<tr>
<td>CHARMAP</td>
<td>Unsupported.</td>
</tr>
<tr>
<td>CHARSET</td>
<td>Supported. This parameter is supported in parameter files and include/obey files. Veridata does not process GLOBALS files.</td>
</tr>
<tr>
<td>COLMATCH</td>
<td>Supported.</td>
</tr>
</tbody>
</table>
### 6.4 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.

<table>
<thead>
<tr>
<th>GoldenGate Parameters</th>
<th>Veridata Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMENT</td>
<td>--</td>
</tr>
<tr>
<td>DICTIONARY</td>
<td></td>
</tr>
<tr>
<td>EXPANDDDL</td>
<td></td>
</tr>
<tr>
<td>EXCLUDEWILDCARDOBJECTSONLY</td>
<td></td>
</tr>
<tr>
<td>EXTRACT</td>
<td></td>
</tr>
<tr>
<td>EXTRAIL</td>
<td></td>
</tr>
<tr>
<td>FILE</td>
<td>TABLE</td>
</tr>
<tr>
<td>INCLUDE</td>
<td></td>
</tr>
<tr>
<td>INSERTALLRECORDS</td>
<td></td>
</tr>
<tr>
<td>INSERTDELETES</td>
<td>NOINSERTDELETES</td>
</tr>
<tr>
<td>INSERTUPDATES</td>
<td>NOINSERTUPDATES</td>
</tr>
<tr>
<td>MACRO</td>
<td></td>
</tr>
<tr>
<td>MACROCHAR</td>
<td></td>
</tr>
<tr>
<td>MAP</td>
<td></td>
</tr>
<tr>
<td>MAP EXCLUDE</td>
<td></td>
</tr>
<tr>
<td>OBEY</td>
<td></td>
</tr>
<tr>
<td>REPLICAT</td>
<td></td>
</tr>
<tr>
<td>RMTTRAIL</td>
<td></td>
</tr>
<tr>
<td>SCHEMAEXCLUDE</td>
<td></td>
</tr>
<tr>
<td>SOURCECATALOG</td>
<td></td>
</tr>
<tr>
<td>TABLE</td>
<td>MAP</td>
</tr>
<tr>
<td>TABLE EXCLUDE</td>
<td></td>
</tr>
<tr>
<td>UPDATEDELETES</td>
<td>NOUPDATEDELETES</td>
</tr>
<tr>
<td>UPDATEINSERTS</td>
<td>NOUPDATEINSERTS</td>
</tr>
<tr>
<td>USEANSISQLQUOTES</td>
<td>NOUSEANSISQLQUOTES</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Veridata Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>TARGET</td>
<td>Supported.</td>
</tr>
<tr>
<td>COLMAP</td>
<td>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 <code>USEDEFAULTS</code> keyword is supported. The <code>BINARYINPUT</code> keyword is ignored.</td>
</tr>
<tr>
<td>COLS</td>
<td>Supported. Results in an explicit column list in the generated compare pair configuration.</td>
</tr>
<tr>
<td>COLSEXCEPT</td>
<td>Supported. If an explicit column mapping does not exist, this results in system mapped columns with a list of omitted columns.</td>
</tr>
<tr>
<td>COORDINATED</td>
<td>Ignored.</td>
</tr>
<tr>
<td>DICTIONARY</td>
<td>Supported for NSK.</td>
</tr>
<tr>
<td>TARGETDICT</td>
<td>Supported for NSK.</td>
</tr>
<tr>
<td>DEF</td>
<td>Supported for NSK.</td>
</tr>
<tr>
<td>TARGETDEF</td>
<td>Supported for NSK.</td>
</tr>
<tr>
<td>EVENTACTIONS</td>
<td>Ignored.</td>
</tr>
<tr>
<td>EXCEPTIONONLY</td>
<td>Unsupported.</td>
</tr>
<tr>
<td>EXITPARAM</td>
<td>Ignored.</td>
</tr>
<tr>
<td>FETCHBEFOREFILTER</td>
<td></td>
</tr>
<tr>
<td>FETCHCOLS</td>
<td>FETCHCOLSEXCEPT</td>
</tr>
<tr>
<td>FETCHMODCOLS</td>
<td>FETCHMODCOLSEXCEPT</td>
</tr>
<tr>
<td>FILTER</td>
<td>Ignored.</td>
</tr>
<tr>
<td>GETBEFORECOLS</td>
<td>Ignored.</td>
</tr>
<tr>
<td>HANDLECOLLISIONS</td>
<td>NOHANDLECOLLISIONS</td>
</tr>
<tr>
<td>INSERTALLRECORDS</td>
<td>Unsupported.</td>
</tr>
<tr>
<td>INSERTAPPEND</td>
<td>NOINSERTAPPEND</td>
</tr>
<tr>
<td>KEYCOLS</td>
<td>Supported.</td>
</tr>
<tr>
<td>MAPEXCEPTION</td>
<td>Ignored.</td>
</tr>
</tbody>
</table>
### Table 6–4 (Cont.) Map and Table Statement Handling

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Veridata Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPERROIR</td>
<td>Ignored.</td>
</tr>
<tr>
<td>RESOLVECONFLICT</td>
<td>Ignored.</td>
</tr>
<tr>
<td>SQLEXEC</td>
<td></td>
</tr>
<tr>
<td>SQLPREDICATE</td>
<td></td>
</tr>
<tr>
<td>THREAD</td>
<td>Ignored.</td>
</tr>
<tr>
<td>THREADRANGE</td>
<td>Ignored.</td>
</tr>
<tr>
<td>TOKENS</td>
<td>Ignored.</td>
</tr>
<tr>
<td>TRIMSPACES</td>
<td>NOTRIMSPACES</td>
</tr>
<tr>
<td>TRIMVARSPACES</td>
<td>NOTRIMVARSPACES</td>
</tr>
<tr>
<td>WHERE</td>
<td>Ignored.</td>
</tr>
</tbody>
</table>
This chapter describes parameters that adjust different aspects of the sort memory configuration when using server-side sorting.

This chapter includes the following sections:

- Overview of the Server Memory
- Estimating Memory Usage
- How to Set a Parameter
- Parameter Descriptions

### 7.1 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.
7.2 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_comparison_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_comparison_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_comparison_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.

7.3 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](http://support.oracle.com).

7.4 Parameter Descriptions

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

- **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
- Server Parameters
- Parameters for Configuring SSL Communication
- Parameters for Veridata Command-Line Utility
- Parameters for Report File Encryption
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_sortBuffers 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.concurrent.writers number

Example

server.concurrent.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.concurrent.readers number

Example

server.concurrent.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
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_fromManagedServer**

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

This appendix describes how to move an Oracle GoldenGate Veridata installation from a test environment (a source environment) to a production environment (a target 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

A.1 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.

For more information about using these scripts, see ”Common Procedures for Moving to a Target Environment” in Administering Oracle Fusion Middleware.

A.2 Additional Steps for Moving Oracle GoldenGate Veridata Repository

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

A.2.1 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:
Additional Steps for Moving Oracle GoldenGate Veridata Repository

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>export</td>
<td>Indicates that the script is exporting Veridata data from the test environment.</td>
</tr>
<tr>
<td>import</td>
<td>Indicates that the script is importing Veridata data to the production environment.</td>
</tr>
<tr>
<td>wluser cmd_user</td>
<td>Specifies the administrative WebLogic Server user.</td>
</tr>
<tr>
<td>exportPassword</td>
<td>Indicates that the script is exporting the cmd_user password.</td>
</tr>
</tbody>
</table>

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

A.2.2 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 = 'S\PROD.$PDATA' || SUBSTR(SRC_TABLE_NAME, 12) Where SRC_TABLE_NAME like 'S\TEST.TDATA%'`
  Update `COMPARE_PAIRS` set `TARG_TABLE_NAME = 'T\PROD.$PDATA' || SUBSTR(TARG_TABLE_NAME, 12) Where TARG_TABLE_NAME like 'T\TEST.TDATA%'`

- **For SQL Server:**
  Update `COMPARE_PAIRS` set `SRC_TABLE_NAME = 'S\PROD.$PDATA' + SUBSTRING(SRC_TABLE_NAME, 12, LEN(SRC_TABLE_NAME) - 12) Where SRC_TABLE_NAME like 'S\TEST.TDATA%'`
  Update `COMPARE_PAIRS` set `TARG_TABLE_NAME = 'T\PROD.$PDATA' + SUBSTRING(TARG_TABLE_NAME, 12, LEN(TARG_TABLE_NAME) -12) Where TARG_TABLE_NAME like 'T\TEST.TDATA%'`

**A.2.3 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:
Additional Steps for Moving Oracle GoldenGate Veridata Repository

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:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration SYSTEM "http://@VeridataServerHost@:@veridataServerPort@/veridata/configuration.dtd">
<configuration validation="required">
  <connection name="@connectionName@" 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:

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

Start the Veridata Agent after making these changes.
This appendix provides the contents of the following 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 Section 5.3, "Configuration File Element Reference".

### B.1 Sample Configuration File

This section shows the contents of a sample configuration file. For more details about each element in this configuration file, see Section 5.3, "Configuration File Element Reference".

```xml
<?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>
  <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>
  </connection>
  <connection name="nskScriptingSource" host="gg-xxxx.us.company.com" port="9999="/>
  <connection name="nskScriptingTarget" host="gg-xxxx.us.company.com" port="9999="/>
</configuration>
</configuration>
</description>
</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>
  <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>
  </connection>
  <connection name="nskScriptingSource" host="gg-xxxx.us.company.com" port="9999="/>
  <connection name="nskScriptingTarget" host="gg-xxxx.us.company.com" port="9999="/>
</configuration>
</description>
</xml version='1.0' encoding='UTF-8'>

<group name="column-mapping" source-conn="sqlScriptingSource" target-conn="sqlScriptingTarget">
  <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)="/>
</configuration>
</description>
```
<![CDATA[
This group has various types of column mapping specifications.
]]>
</description>
<compare-pair source-table="SYSMAPPING1" name="excludeCol6">
  <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[...
]]>
</query></source-config>
</delta-config>
</compare-pair>
</group>
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">
<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>
</compare-pair>
</group>
Sample Configuration File

<sql-partition name="part2" side="source">
  <![CDATA[
     col2 > 20
  ]]>  
</sql-partition>

<sql-partition name="part2" side="target">
  <![CDATA[
     col3 > 30
  ]]>  
</sql-partition>

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

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

<group name="compare-pair-with-pkey" source-conn="sqlScriptingSource" target-conn="sqlScriptingTarget">
  <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 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="$FSS04.FSSVDCT" record="ENSSRC-REC">
      <expandddl
        redefined-columns ="include"
        resolvedups="appendAlphaIndex"
        expandGroupArrays="false"
        ddl-separator="underscore"
        zero-fill-length="1"
        fix-long-names="false"
        max-col-name-len="110"/>
    </enscribe-info>
</compare-pair>
</group>
<enscribe-info side="target"
dictionary="$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>
</compare-pair>
</group>

<group name="include-exclude-filter" source-conn="sqlScriptingSource" target-conn="sqlScriptingTarget" source-catalog="" target-catalog="">
<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>

<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="" name="userDefinedCols"> <!-- exclude col6 -->
<column source-name="COL5" target-name="COL5"/>
<column source-name="COL6" target-name="COL6"/>
</compare-pair>
<compare-pair source-table="SYSMAPPING4" target-table="" name="userDefinedCols"> <!-- exclude col6 -->
<column source-name="COL5" target-name="COL5"/>
<column source-name="COL6" target-name="COL6"/>
</compare-pair>
</group>

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

B-6  Administering Oracle GoldenGate Veridata
<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="777777" />
<param name="write-success-rpt" value="false" />
<param name="disable-trigger" value="true" />
</repair>
</profile>
</configuration>
This appendix provides the profile parameter decryption for use in configuring profiles used with the Oracle GoldenGate Veridata import and export utilities.

For more information about the parameters used in this configuration file, see Section 5.3, "Configuration File Element Reference".

### C.1 General (profile-general)

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

### C.2 Sorting Method (sorting-method)

<table>
<thead>
<tr>
<th>Name on UI</th>
<th>Name to be used in configuration XML file</th>
<th>Default Values</th>
<th>Allowed Values</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sort Data Using</td>
<td>sort-method</td>
<td>database</td>
<td>server, database</td>
<td>Enumeration</td>
</tr>
<tr>
<td>Maximum Memory Usage (MB)</td>
<td>sort-max-memory</td>
<td>50</td>
<td>1 to 100000</td>
<td>int</td>
</tr>
<tr>
<td>Temporary Storage Directory for Source Data</td>
<td>sort-src-temp-dir</td>
<td>string</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temporary Storage Directory for Target Data</td>
<td>sort-tar-temp-dir</td>
<td>string</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### C.3 Initial Compare (initial-compare)

<table>
<thead>
<tr>
<th>Name on UI</th>
<th>Name to be used in config XML file</th>
<th>Default Values</th>
<th>Allowed Values</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Concurrent Comparison Threads</td>
<td>max-thread</td>
<td>4</td>
<td>1 to 20</td>
<td>Int</td>
</tr>
<tr>
<td>Terminate when Maximum Records Out-Of-Sync</td>
<td>max-oos-record</td>
<td>100000</td>
<td>0 to 100000000</td>
<td>int</td>
</tr>
<tr>
<td>Output Out-Of-Sync Record Details to Report File</td>
<td>output-oos-rpt</td>
<td>false</td>
<td>true, false</td>
<td>boolean</td>
</tr>
<tr>
<td>Update Report file Every (seconds)</td>
<td>update-rpt-second</td>
<td>0</td>
<td>0 to 10000000</td>
<td>int</td>
</tr>
<tr>
<td>Update Report file Every (records)</td>
<td>update-rpt-record</td>
<td>0</td>
<td>0 to 1000000000</td>
<td>int</td>
</tr>
<tr>
<td>Limit Number of Input Rows</td>
<td>limit-input-row</td>
<td>0</td>
<td>0 to 1000000000</td>
<td>int</td>
</tr>
<tr>
<td>Source Oracle optimizer hint</td>
<td>src-oracle-hint</td>
<td>string</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target Oracle optimizer hint</td>
<td>tar-oracle-hint</td>
<td>string</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generate Messages</td>
<td>rpt-msg</td>
<td>none</td>
<td>none, info, warning, both</td>
<td>Enumeration</td>
</tr>
<tr>
<td>Generate Warning Messages For Out-Of-Sync Rows After (differences)</td>
<td>rpt-warn-msg-threshold</td>
<td>50</td>
<td>0 to 10000000</td>
<td>int</td>
</tr>
<tr>
<td>Use Static Listening Port For Agent During Row Hash On Source (0 to use dynamic port list)</td>
<td>src-agent-static-port</td>
<td>0</td>
<td>0 to 65535</td>
<td>int</td>
</tr>
<tr>
<td>Use Static Listening Port For Agent During Row Hash On Target (0 to use dynamic port list)</td>
<td>tar-agent-static-port</td>
<td>0</td>
<td>0 to 65535</td>
<td>int</td>
</tr>
<tr>
<td>Source Process Name Starting With (Must start with '$', followed by two letters, and end with '<em>'. Example: $AA</em>)</td>
<td>src-nsk-name</td>
<td>string</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source Process CPU Number</td>
<td>src-nsk-cpu</td>
<td>-1</td>
<td>-1 to 16</td>
<td>int</td>
</tr>
<tr>
<td>Source Process Priority</td>
<td>src-nsk-priority</td>
<td>0</td>
<td>0 to 10000000</td>
<td>int</td>
</tr>
</tbody>
</table>
### C.4 Confirm-Out-Of-Sync (confirm-out-of-sync)

<table>
<thead>
<tr>
<th>Name on UI</th>
<th>Name to be used in configuration XML file</th>
<th>Default Values</th>
<th>Allowed Values</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform Confirm Out-Of-Sync Step</td>
<td>coos-enable</td>
<td>true</td>
<td>true, false</td>
<td>boolean</td>
</tr>
<tr>
<td>Run Concurrently With Initial Compare</td>
<td>coos-concurrent</td>
<td>true</td>
<td>true, false</td>
<td>boolean</td>
</tr>
<tr>
<td>Confirm-Out-Of-Sync Batch Size</td>
<td>Batch-size</td>
<td>10</td>
<td>1 to 100</td>
<td>int</td>
</tr>
<tr>
<td>Delay Confirm-Out-Of-Sync By (seconds)</td>
<td>coos-delay</td>
<td>0</td>
<td>0 to 1000000</td>
<td>int</td>
</tr>
<tr>
<td>Terminate when Maximum Records Out-Of-Sync</td>
<td>coos-max-oos</td>
<td>100000</td>
<td>0 to 10000000</td>
<td>int</td>
</tr>
<tr>
<td>Output Out-Of-Sync Record Details to Report File</td>
<td>coos-output-oos-rpt</td>
<td>false</td>
<td>true, false</td>
<td>boolean</td>
</tr>
<tr>
<td>Update Report file Every (seconds)</td>
<td>coos-output-rpt-seco</td>
<td>0</td>
<td>0 to 10000000</td>
<td>int</td>
</tr>
<tr>
<td>Update Report file Every (records)</td>
<td>coos-output-rpt-recod</td>
<td>0</td>
<td>0 to 10000000</td>
<td>int</td>
</tr>
<tr>
<td>Source Oracle optimizer hint</td>
<td>coos-src-oracle-hint</td>
<td>string</td>
<td></td>
<td>string</td>
</tr>
<tr>
<td>Target Oracle optimizer hint</td>
<td>coos-tar-oracle-hint</td>
<td>string</td>
<td></td>
<td>string</td>
</tr>
<tr>
<td>Generate Messages</td>
<td>rpt-msg</td>
<td>none</td>
<td>none, info, warning, both</td>
<td>Enumeration</td>
</tr>
<tr>
<td>Generate Warning Messages For Out-Of-Sync Rows After (differences)</td>
<td>rpt-warn-msg-threshold</td>
<td>50</td>
<td>0 to 1000000</td>
<td>int</td>
</tr>
<tr>
<td>Use Static Listening Port For Agent During Row Hash On Source (0 to use dynamic port list)</td>
<td>src-agent-static-port</td>
<td>0</td>
<td>0 to 65535</td>
<td>int</td>
</tr>
<tr>
<td>Name on UI</td>
<td>Name to be used in configuration XML file</td>
<td>Default Values</td>
<td>Allowed Values</td>
<td>Type</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
<td>----------------</td>
<td>---------------</td>
<td>--------</td>
</tr>
<tr>
<td>Run Repair Automatically after Compare</td>
<td>repair-after-compare</td>
<td>False</td>
<td>True,false</td>
<td>boolean</td>
</tr>
<tr>
<td>Repair batch size</td>
<td>batch-size</td>
<td>10</td>
<td>1 to 100</td>
<td>int</td>
</tr>
<tr>
<td>Repair transaction size</td>
<td>txn-size</td>
<td>1</td>
<td>0 to 100</td>
<td>int</td>
</tr>
<tr>
<td>Concurrent Repair Operations</td>
<td>concurrent-operation</td>
<td>1</td>
<td>1 to 100</td>
<td>int</td>
</tr>
<tr>
<td>Check changed values</td>
<td>check-change-value</td>
<td>true</td>
<td>true,false</td>
<td>boolean</td>
</tr>
<tr>
<td>Terminate when maximum repair warnings</td>
<td>terminate-max-warn</td>
<td>10000</td>
<td>0 to 2147483647</td>
<td>Int</td>
</tr>
<tr>
<td>Write Repair Success Messages to Report</td>
<td>write-success-rpt</td>
<td>true</td>
<td>true,false</td>
<td>boolean</td>
</tr>
<tr>
<td>Disable DB Triggers Session Based</td>
<td>disable-trigger</td>
<td>false</td>
<td>true,false</td>
<td>boolean</td>
</tr>
<tr>
<td>Disable DB Triggers Session Based</td>
<td>disable-trigger</td>
<td>false</td>
<td>true,false</td>
<td>boolean</td>
</tr>
</tbody>
</table>